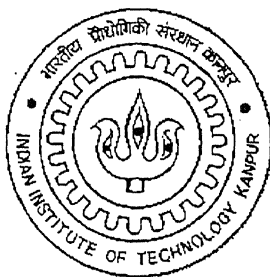


Tolerance Relations For Signal Processing

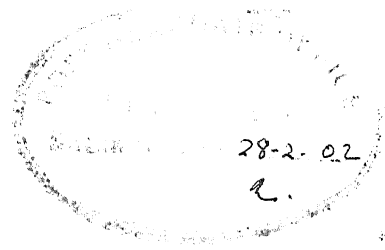
*A Thesis Submitted
in Partial Fulfillment of the Requirements
for the Degree of
Master of Technology.*

By
Santanu Bijoy Majumder



**Department of Electrical Engineering
Indian Institute of Technology Kanpur**

February 2002



Certificate

This is to certify that the work contained in this thesis entitled "*Tolerance Relations For Signal Processing*", by Santanu Bijoy Majumder, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

February 2002

(K.S. Venkatesh)
Assistant Professor.
Department of Electrical Engineering,
Indian Institute of Technology,
Kanpur.

26 APR 2002

/EE

गुरुषोत्तम काशीनाथ केवकर पुस्तकालय

भारतीय प्रौद्योगिकी संस्थान, रायपुर

अवधि क्र० A.....139574.....



A139574

Acknowledgement

I acknowledge a number of individuals who have contributed in different ways to the completion my thesis work. At the outset I am very much indebted to my thesis Supervisor Dr. K.S. Venkatesh. For his invaluable guidance, constant encouragement and freedom throughout the course of my work. His teaching was not only a pleasant experience, it was an introduction to a new world. Words are not enough to express my gratitude towards him.

I would also like to express my deep sense of gratitude to my teachers Dr.V.P.Sinha, Dr.Sumana Gupta, Dr.P.K.Kalra, Dr.K.E.Hole and last but not least Dr P.K.Sarkar for teaching me various courses during my course work.

I am also very much thankful to my friends Sashikant, harshavardhan, K.murlimohan, S.Mishra and Anil for various help and maintaining a good environment in DSP lab.

Abstract

The idea of a tolerance relation evolved many years ago. But it has not been applied for DSP problems before. We study this mathematical tool for DSP problems. The report covers the notion of tolerance relations; its application in signal representation and reconstruction, for standard DSP problems like detection of video shot change, and the detection of the appearance of a new object in a scene. The proposed method detects objects using a device we term a *tolerance meter*, then builds an object-to-object correspondence between consecutive frames using tolerance criteria and finally also detects changes using a tolerance relation. We also study the application of tolerance to detect textural objects. The algorithm we propose shows promise.

List of Tables

3.1	Correspondence between the first frame and the fourth frame	44
5.1	The results with various values of α .	63
5.2	The results with various values of β .	63
5.3	The distance matrix considering intensity.	66
5.4	The distance matrix considering size.	66
5.5	The established correspondence.	68
5.6	Change detection of sequence1.	71
5.7	Change detection of sequence2.	72
5.8	Change detection of sequence3.	73
5.9	Change detection of sequence4.	74
5.10	Change detection of sequence5.	75

List of Figures

1.1. Transitive closure.	4
2.1. Tolerance meters.	13
2.2. Input and Output of tolerance meter	14
2.3. Sampled signal	15
2.4. Output of tolerance meter with very small preset.	15
2.5. Tolerance meter output with different preset	15
2.6. Adjacent neighbors	17
2.7. Reconstruction of 1D signal.	18
2.8. Reconstruction of 2D signals.	20
2.9. The difference image for one color component.	20
2.10. Reconstructed components.	21
2.11. Possible reconstructed images....	23
3.1. Identified objects using tolerance classes.	29
3.2. Identified objects using tolerance bases.	31
3.3. Identified objects using modification	33
3.4. Identified objects using transitive closure.	34
3.5. Block Diagram to determine object-to-object correspondence.	39
3.6. Frames used to establish object to object correspondence	40
3.7. Objects of the fourth frame.	41

4.1. Example of textural objects.	48
4.2. The test image used for texture segmentation.	52
4.3. The whole low frequency parts of the block DCT2 output.	54
4.4. Only the first element of the low frequency part	54
4.5. Partitions with preset 2701.	55
4.6. Partitions with preset 1890.	55
4.7. Partitions induced with the results of first and second channels.	55
4.8. Partitions with preset 890.	56
4.9. Partitions induced with all tolerance meter outputs.	56
4.10. Identified partitions after smoothing.	57
4.11. Identified objects.	59
4.12. Identified objects in the original test image.	60.
5.1. Identified Objects.	64
5.2. Identified objects.	65
5.3. Two consecutive frames used for setting the values of γ and δ	67
5.4. The flow chart of the proposed algorithm	69
5.5. The test sequences.	70

Contents

1. Introduction.	1
1.1 The Mathematical preliminaries of tolerance relation.	2
2. The signal space.	13
2.1. The Tolerance meter.	14
2.2. Reconstruction of the input signal from the tolerance meter output	14
2.3. Reconstruction of Images.	19
2.4. The results	20
3. Application of tolerance relation to identify reasonable change	25
3.1. What is reasonable change?	25
3.2. Definition of the problem.	26
3.3. Detection of inter-shot change: fixed camera case.	27
3.4. Detect of inter-shot changes: moving camera case.	46
3.5. Detection of the appearance of new object in a scene.	46
4. Texture Identification.	48
4.1. Quantifying the differences in textures	48
4.2. Texture Segmentation.	49
5. Algorithms and results.	62
6. Conclusion.	76

Section 1

Introduction

The idea of a tolerance relation came long time ago. It is related to the idea of closeness. There are many practical situations we find in which the action we need to take, or the interpretation we make is decided by the closeness or similarity relationship between objects. This is particularly true in the context of image and video processing. We attempt in what follows to develop a new family of tools for various signal processing operations that are based on the tolerance concept. For example, there are many developed tools available for image segmentation: we propose a fundamentally nonlinear tool for this operation that can segment images at places where there is a significant change of appearance, while at the same time remaining oblivious to gradual change. The beauty of the tolerance relation is that it can be used to determine closeness between objects in any context, if they belong to the same set. So, for video sequences, it can be used to determine closeness from the pixel level to the frame level.

We explored the problem of detecting changes in a scene while at the same time, neglecting certain minor changes that we considered irrelevant, like the slow movement of the loose ends of any object and changes of brightness. The neglecting of minor changes are even more important than merely detecting the ‘major’ changes. This problem gave rise to the notion of a *tolerance meter*, a notion that plays a central role in this thesis.

The thesis is organized as follows. First, we introduce the notion of tolerance. In Section 2, we apply it to one- and two-dimensional signal representation. In Section 3 we apply the theory to change detection in video. Section 4 covers the study of textural objects. Section 5 presents our algorithm and results. Section 6 is the conclusion and covers further scope of using tolerance relations. The mathematical preliminaries of tolerance are taken from [1] and fundamental concepts of images analysis from [2].

1.1 The Mathematical Preliminaries of Tolerance Relations

The notions of Relations

A relation is something that holds between objects. Without objects, it does not exist.

A relation may be defined between two objects (binary relation) or more, (n-ary relation). The relation “to be the sum of “ is an example of a 3-ary relation (x, y, z)

$$X+Y=Z$$

Our focus will be on binary relations. There are several ways to represent a relation. For algebraic representation let's consider a given finite set M ; a relation is to be defined on the members of this set. We pick up all pairs of the members of M in the form $\langle x, y \rangle$ $x, y \in M$ between which the relation holds and constitute a set. One way to denote the set of all ordered pairs is the Cartesian product $M \times M$. The binary relation R will be some of the ordered pairs, or a subset of $M \times M$. So R will be the subset such that if xRy $\langle x, y \rangle \in R$.

So $R \subseteq M \times M$. The relational structure will be expressed as $\langle R, M \rangle$ where M is the support of R and R is the graph $\langle R, M \rangle$.

The geometric way of representing a binary relation on a finite set is by an oriented graph. Here each element of the support is represented by a vertex and if xRy , a directed line is drawn from x to y .

A special case of a relation is a function. A little consideration will clarify this. Let us consider a relation with the support M such that for every $x \in M$ there exists only one element $y \in M$ for which xAy holds. So by just picking up any element from M , there is produced a corresponding element. This relation is called a function or mapping. This correspondence may be unique. As in function $y = f(x)$ we say y is the value of the function in the point x . Here the dependence of y on x will be expressed as $y = A(x)$ and

it means $y \in M$ is the value of the function A in the element x . As we can plot the y versus x here also it is possible to talk about the graph where both axis are M .

We may generalize the notion of a relation to that of a so-called correspond structure that operates on elements from two different ground sets instead of *i.e.* $x \in M$ & $y \in L$. We will call f a function if for each $x \in M$ when there exists unique y such that $y \in L$, xfy or $y = f(x)$. In relational parlance M is called domain departure and L is called domain of arrival what is usually known as domain and range. Here also we say y as image of x and x as pre-image of y .

In a mapping defined as $f : M \rightarrow L$ each element $x \in M$ has exactly one image and it is not necessary for each $y \in L$ to have a pre-image.

Operations on Relations

Since we are considering relational structures as sets, all the set operations are meaningful on relations.

- 1 Intersection: Given two relational structures (A, M) and (B, M) , the new relation determined by the intersection of the sets A and B will be called $A \cap B$ on M
- 2 Union: Analogously, the union $A \cup B$ of relations A & B will mean the relation determined by the union of the corresponding sets.
- 3 Subset: We can pick $A \subset B$ if the set of pairs A is a subset of B but $A \neq B$
- 4 Inverse Relation: Consider a relational structure (R, M) . The inverse relation structure (R^{-1}, M) is defined by x, y such that if xRy holds then $yR^{-1}x$. So yRx is equivalent to $xR^{-1}y$.
- 5 Product: The relation AB exists if there exists an element $z \in M$ such that xAz and zBy holds for any $x, y \in M$: The product relation AB possesses several algebraic properties.

Lets consider the product AA . Here if xAy and yAz then xAz . It gives us the idea of another important operation transitive closure for a relational structure (A, M) . The transitive closure \hat{A} is defined as the pairs $\langle x, y \rangle$ for which $x\hat{A}y$ hold if there exists a sequence of elements of M : $z_0 = x, z_1 = \dots, z_n = y$ and the relation A holds for all neighbors i.e. $z_0Az_1, z_1Az_2, \dots, z_{n-1}Az_n$.

So it is evident that $A \subseteq \hat{A}$. Now we say $x\hat{A}y$ at least for one n . Using the union operation this fact can be written as

$$\hat{A} = A \cup A^2 \cup A^3 \cup \dots \cup A^n \cup \dots$$

The operation \hat{A} of transitive closure can be expressed in matrix form by the union of the matrices representing the various powers of A . It is however more intuitive to pass from the graph representing the relations A to the graph representing \hat{A} .

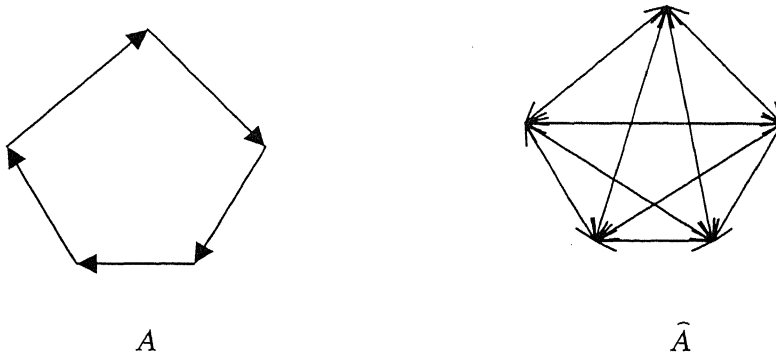


Fig 1.1. Transitive closure.

Some Algebraic Properties of Relations

1. Reflexivity: The relation A is called reflexive if identity relation is a subset of A .

In other words $xAx \quad \forall x \in M$.

2. Anti-reflexivity: The relation A is called anti reflexive if xAx implies $x \neq y$

i.e. $A \cap E = \Phi$ where E is identity mapping.

3. Symmetry: The relation A is called symmetric if $A \subseteq A^{-1}$ i.e. if xAy holds yAx also holds for $x, y \in M$.
4. Asymmetry: The relation is called asymmetric if $A \cap A^{-1} = \Phi$.
5. Anti-symmetry: The relation A is called anti-symmetric if $A \cap A^{-1} \subseteq E$.
6. Transitivity: The relation A is called transitive if $A^2 \subseteq A$, i.e. if xAz and zAy then xAy also holds.

Some definitions:

1. Surjective Mapping: The mapping $\alpha: M \rightarrow L$ is called a surjective mapping if each element y from L has a pre-image. We also say M is mapped onto L .
2. Injective Mapping: The mapping $\alpha: M \rightarrow L$ is called injective if each element $y \in L$ has at most one pre-image.
3. Bijective mapping: If the mapping $\alpha: M \rightarrow L$ is simultaneously surjective and injective it is called a bijective mapping. The sets M, L are said to be equipollent.

The Notion of Equivalence

To introduce the notion of an equivalence relation, we speak of the notion of substitutability or interchangeability of objects in a specific context.

Now let's consider a set of objects M out of which some objects are identical. Let's denote the set of all objects which are interchangeable with x as M_x . So it is obvious that $x \in M_x$. And it is very obvious that the union of all such M_x will coincide to M . i.e.

$M = \bigcup_{x \in M} M_x$. Importantly, these sets either coincide completely or are disjoint.

definition of a partition:

A partition on a set M is a system consisting of non-empty subsets of M $\{M_1, M_2, \dots\}$ such that

1. $M = M_1 \cup M_2 \cup \dots$
2. $M_i \cap M_j = \emptyset$ for $i \neq j$.

These sets are called classes.

definition of equivalence relation:

A relation A on a set M is called an equivalence relation if there exists a partition $\{M_1, M_2, M_3, \dots\}$ of the set M such that the relation xAy holds if and only if x and y belong to the same class M_i of the partition.

The previous definition is made with the aid of partition. From other point of view with the aid of algebraic property equivalence relation is defined: the relation A on a set is called an equivalence relation if it is reflexive, symmetric and transitive.

The consistency between two definitions can be proved easily both ways i.e. if A is reflexive, symmetric and transitive it induces a disjoint partition in the set M . and a disjoint partition always induces a relation with the aforesaid properties.

Difference Between Identity and Resemblance

Identity implies that the given objects are same to the viewer and the partition is clear. But resemblance is a situation where partition is not clear because there is no clear boundary between the classes and one element can belong to more than one class. If one object resembles more than one object, these different objects themselves may not necessarily resemble to each other, because the minor difference may grow enough not to make them resemble. In this situation, there arises the notion of a tolerance relation.

Tolerance Relation

The relation A on a set M is called a tolerance relation if it is reflexive and symmetric.

Its axiomatization is intuitive because each element resembles itself and if one element resembles another, vice versa also holds. And the third property what was true for equivalence relation, does not hold for tolerance relations in general.

Equivalence Relation as a Special Case of Tolerance Relation

It is clear that if a relation is an equivalence, it must be a tolerance relation too, as it satisfies all the tolerance axioms. But a general tolerance relation is not necessarily an equivalence relation.

Tolerance Space: If a tolerance relation τ is defined on a set M then the relational structure (M, τ) is called tolerance space. Another way to represent a tolerance relation other than by the defining axioms is with the aid of a correspondence.

We establish a correspondence $\varphi : M \rightarrow L$, and denote the set of all images of the element x under the correspondence φ by $\Phi(x)$. We consider a new relation A_φ on the set M defined by the condition $x A_\varphi y$ if the elements x and y have a common image i.e. $\Phi(x) \cap \Phi(y) \neq \emptyset$. To establish consistency with the previous definition, we look at the algebraic properties of A_φ .

Property 1. If the correspondence φ is defined on all elements of M , the relation A_φ is reflexive. This is very obvious because $\Phi(x) \cap \Phi(x) = \Phi(x)$ is always non-empty.

Property 2. The relation A_φ is always symmetric. Because $\Phi(x) \cap \Phi(y) = \Phi(y) \cap \Phi(x)$.

Property 3. If the correspondence φ is not defined on any element x in M then it will not be a relative of any element under relation A_φ . This is because $\Phi(x) = \emptyset$. So $\Phi(x) \cap \Phi(y) = \emptyset$.

Property 4. If the correspondence $\varphi: M \rightarrow L$ is a function i.e. for any element $x \in M$, $\Phi(x)$ consists of not more than one element, then the relation A_φ will be transitive and turns out to be an equivalence relation.

This is because, if $x A_\varphi y$ and $y A_\varphi z$ then $\varphi(x) = \varphi(y)$ and $\varphi(y) = \varphi(z)$ and consequently $\varphi(x) = \varphi(z)$ i.e. $x A_\varphi z$.

From the properties of A_φ , it is evident that for an everywhere defined correspondence $\varphi: M \rightarrow L$ on M , a symmetric and reflexive relation A_φ is a tolerance relation. Conversely, it is also possible to go from any arbitrary tolerance relation A_φ to an induced everywhere defined correspondence φ .

Operations on Tolerances

If A and B are tolerances, the following relations: $A \cup B, A \cap B, A^{-1}$ and \bar{A} are also tolerances.

Lemma: If A is a tolerance relation, B is an equivalence relation and $A \subseteq B$, then $\bar{A} \subseteq B$.

It is easy to prove this. Just applying transitive closure both sides will yield the result. The meaning of transitive closure of a tolerance relation is that it is the minimal equivalence relation containing that tolerance. For the product of tolerances, there is a theorem as follows.

Theorem:

In order for the product AB of the two tolerance relations, A and B , to be a tolerance, it is necessary and sufficient that A and B commute. In this case, $AB=A \circ B$.

The symmetrized product $A \circ B = AB \cup BA$

$$(A \circ B)^{-1} = (AB \cup BA)^{-1} = (AB)^{-1} \cup (BA)^{-1} = B^{-1}A^{-1} \cup A^{-1}B^{-1} = BA \cup AB = AB \cup BA = A \circ B.$$

It is also possible to take symmetrized product $A * B = (AB \cap BA)$ and it is easy to show that $A * B$ is tolerance if A and B are tolerances.

It is also worthwhile noting that that for any reflexive relation A the following relations are tolerances: $A \cup A^{-1}, A \cap A^{-1}, A \circ A^{-1}$.

Tolerances Classes

A tolerance relation A_φ in a set M can also be defined in the language of coverings. A system of sets Π is called a covering of the set M if $\bigcup_{A \in \Pi} A \supseteq M$. We make some simple definitions and present some statements without proof.

Definition of pre-class: A set $L \subseteq M$ is called a pre-class in the tolerance space $\langle M, \tau \rangle$ if any two of its elements x and y are tolerant.

Definition of class: A set $K \subseteq M$ is called a tolerance class in $\langle M, \tau \rangle$ if K is the maximal pre-class there.

Lemma: In order that the elements x and y be tolerant, it is necessary and sufficient that there exists a pre-class containing both elements.

Lemma: Every pre class is contained in at least one class K.

Corollary: The system of tolerance classes forms a covering of the set M.

Lemma: In order that the elements x and y be tolerant, it is necessary and sufficient that there exists a class containing both elements. We define S_H as the set of all non-empty subsets of H.

Theorem:

Let $\langle M, \tau \rangle$ be an arbitrary tolerance space, and let H be the set of all its tolerance classes. Then there exists a mapping $\varphi: M \rightarrow S_H$ such that elements of M are tolerant if and only if their images are tolerant in S_H .

Proof of this theorem is not difficult. First we define the mapping as follows: for any $x \in M$, $H(x)$ is the set containing all classes containing x. So if $x\tau y$ then $H(x)$ and $H(y)$ have at least one class in common. So we can say $H(x)$ and $H(y)$ are also tolerances in the space S_H with a defined tolerance relation.

For finite cases we can also take the mapping as $\varphi: M \rightarrow S_p$ where p is the number of tolerance classes in $\langle M, \tau \rangle$ which assigns to each element x the set of numbers of the classes containing it;

$$x \rightarrow \{n_1, n_2, \dots, n_k\} \text{ Here } n_i \leq p.$$

Now consider the everywhere defined correspondence $\varphi: M \rightarrow H$, which assigns to each element $x \in M$, all classes in which it occurs. So $x\tau y$ is equivalent to x and y having a common image in H.

Theorem:

An arbitrary tolerance relation τ in a set M can be given as a relation A_φ , with the aid of some everywhere defined correspondence $\varphi: M \rightarrow H$.

Definition of Basis: A collection $H_B = \{K^1, K^2, \dots\}$ of classes in a tolerance space $\langle M, \tau \rangle$ is called a basis if

1. For every tolerant pair x and y , there exists a class $K^i \in H_B$, containing both the elements: $x \in K^i, y \in K^i$;
2. The deletion of any class from H_B leads to the loss of this property.

Now using the idea of basis there is a theorem:

Theorem:

Let $\langle M, \tau \rangle$ be an arbitrary tolerance space in which H_B is a basis. Then there exists a mapping $\varphi: M \rightarrow S_{H_B}$, such that elements of M are tolerant if and only if their images are tolerant in S_{H_B} . The proof of this theorem is also very evident, as it follows very closely the earlier theorem regarding the map $\varphi: M \rightarrow S_{H_B}$.

Lemma: A tolerance relation is an equivalence relation if and only if the tolerance classes do not intersect each other.

Definition of kernel: Let $\langle M, \tau \rangle$ be a tolerance space. A set $L \subseteq M$ is called a kernel, if there exists a collection of classes K^1, K^2, \dots , such that L is the set of all elements from M , each of which occurs in all these, and only these, classes.

Kernels are the pre-images under the mapping $\varphi: M \rightarrow S_{H_B}$. In fact the kernel (K^1, K^2, \dots) consists of all those elements x , whose image $\varphi(x)$ is precisely this set of tolerance classes: $\{K^1, K^2, \dots\}$.

Now let's define another relation in M by the condition: $x \theta y$ if $T_x = T_y$ where T_x is the set consisting all elements tolerant to x in the tolerance space $\langle M, \tau \rangle$.

Lemma: In order for the relation $x\theta y$ to hold, it is necessary and sufficient that x and y lie in one and the same kernel (K^1, K^2, \dots) .

Corollary: The relation θ is equivalence, and the non-empty kernels serve as equivalence classes for θ . This conclusion is obvious because $(K^1, K^2, \dots) \subseteq K^1 \cap K^2 \cap \dots$.

In the case of equivalence, classes do not intersect and each kernel coincides with its tolerance class: $(K) = K$. and for any $x \in (K)$, $T_x = (K)$.

Definition of Kernel free: A tolerance space $\langle M, \tau \rangle$ is called kernel free, if each of its kernels consists of not more than one element.

Theorem:

Let $\langle M, \tau \rangle$ be a kernel-free tolerance space, and H the set of all its tolerance classes. Then there exists an injective mapping $\varphi: M \rightarrow S_H$, such that elements of M are tolerant if and only if their images are tolerant in S_H .

Lemma: The partition of the set M into kernels with respect to the basis H_B coincides with the partition of M into ordinary kernels.

Theorem:

If the tolerance space $\langle M, \tau \rangle$ has a finite basis H_B , then the set of all tolerance classes in $\langle M, \tau \rangle$ is finite.

ction 2

The Signal Space

in this section we discussed, how tolerance relation could be applied in the space where each signal is an element.

In order to apply the theory of tolerance relations in the space of signals, we require a classification upon the signal space with respect to some feature correspondence on the basis of which it will be determined whether two signals are tolerant or not. This corresponds to having a black box that extracts the desired feature and maps the signals to another domain where two signals would result in the same output if they belong to some common tolerance class (are tolerant).

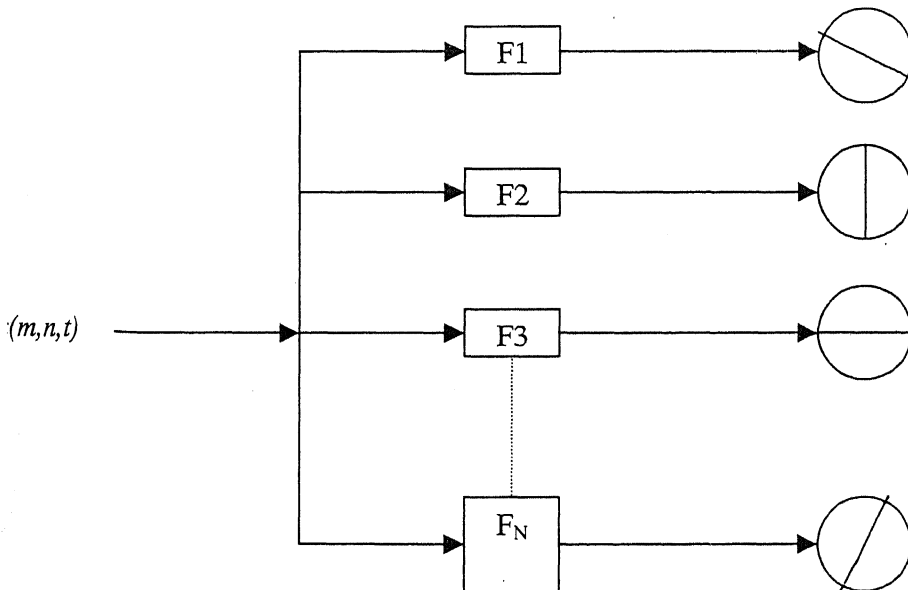


Fig.2.1 Tolerance meters.

Here we depicted a collection of *tolerance meters* which get the signals as input and give an answer of yes or no so that each meter divides the whole signal space into two classes.

The Tolerance Meter

Now the question is what a tolerance meter will exactly do? Each tolerance meter implements a certain correspondence. Selection of the feature will depend upon the particular problem. The following is a simple example of the use of a tolerance meter.

Any digital one-dimensional signal amounts to an array of numbers. So let's consider a signal of fifty samples: an array of length fifty. Now our tolerance meter will have a *preset* whose value will affect its functioning as follows. For the signal $x(t)$ each sample will be compared with the next sample. If the absolute difference is less than the tolerance meter preset value then at that position it will assign one as its output, else it will assign zero. That way we will get an output signal of length forty-nine consisting of zeros and ones. At each sample position, we are effectively dividing the whole signal space into two classes. One is corresponding to yes and another to no. The combination of these outputs gives a narrower partitioning than each does, on its own.

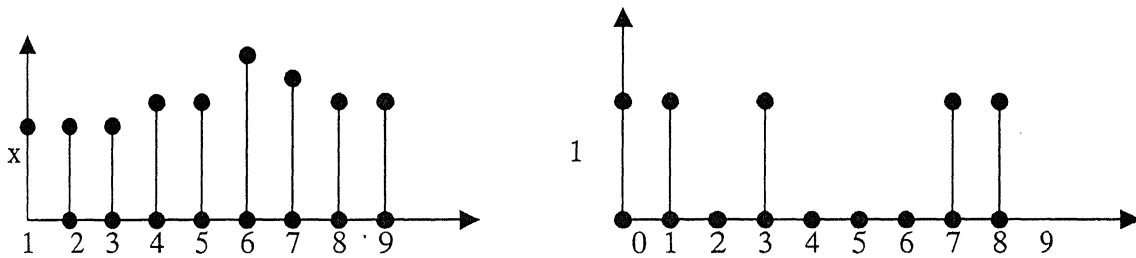


Fig.2.2 Input and output of tolerance meter.

We can also think of the comparison of the samples with the second next leaving the first one. This tolerance meter output will be called a second order output, and on the same principle, we can implement tolerance meters to produce any n -th order output.

Reconstruction of the Input Signal from the Tolerance Meter Output

Here, we try to reconstruct the input signal from the meter output. But in the output we are only getting zeros and ones. So from only one output it is not possible to get back. Since the output of the tolerance meter comes from the differences, for the reconstruction we need to get the

difference of consecutive samples. But the output only consists of zero and one, so it can't provide the absolute difference. The following idea will give an indication how to get the differences.

Let us take a signal like the following.

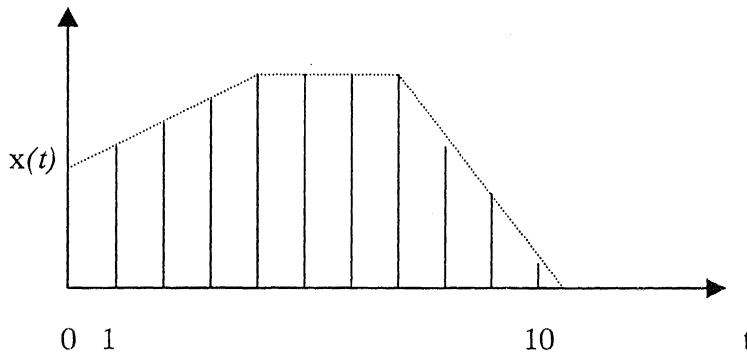


Fig.2.3 Sampled signal

There are three slopes in this signal. First part is with a slow positive slope, second part with no slope, and the third part with high negative slope. If the tolerance meter preset value is smaller than the value of smaller slope then the output will come like the following.

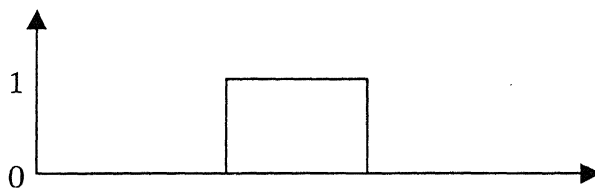


Fig. 2.4 Output of tolerance meter with very small preset.

If the preset value is set to be in-between the smaller and larger slope value then the output will be of the shape shown below to the left. Finally, if the preset value is greater than the larger slope the output will look like the diagram to the right.

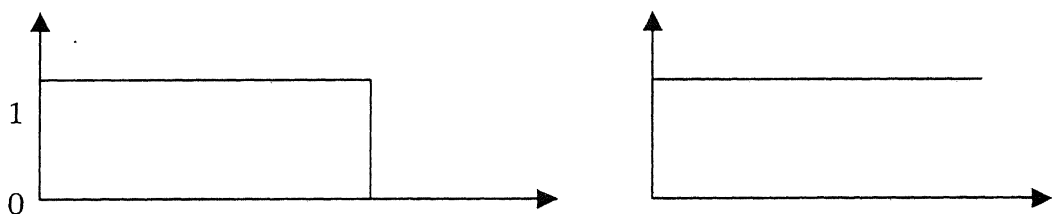


Fig.2.5 Tolerance meter output with different preset.

Now the idea to get the difference from these results is that if we slowly vary the preset value from zero to the maximum possible difference then the output will start to vary from zero to one for each sample at different times when the preset will be nearest to the difference value. This procedure will yield the difference output that is nothing but the consecutive sample differences. The maximum possible error we can get in the difference is the step size with which we increase the preset value.

If we know the difference array, we will face another problem during reconstruction because we are getting the differences in absolute values. So, if the first sample is given to us there are two possibilities for the second sample. One is below the first one and another is above the first one. So if we proceed considering the possible signals having same differences array the number will increase two times. And the number will be too great to tackle. One way to alleviate this problem is to look at the second order differences, because all the possible inputs can't have the same second order difference. So we will pick up those possibilities that also satisfy the second order differences. Now we have a means to check and pick up the right one from the two possibilities for each sample.

Here another question arises. Do these two suffice to take us to the unique input signal or do we need higher order differences?

Let us consider a signal with the following sample values:

[5 3 6 8 2 4 6 6 4 3 9 4]; Here the first order difference is [2 3 2 6 2 2 0 2 1 6 5] and the second order difference is [1 5 4 4 4 2 2 3 5 1]. Now since there is one zero in the first order difference. So when we go to reconstruct the ninth sample (4), we get first order difference 2 and second order difference 2. So again two possibilities arises. So here we required third order difference.

So both the first order and second order differences used for calculating a particular sample turn out to be same, then again two possibilities will arise. That necessitates appeal to third order differences. How can we know when such a case will arise? Such a case will arise if there is any zero in the first order difference array. But the third order difference will also fail if

there is two consecutive zeroes is the first order difference. Then we need to go for even higher order differences. But how far should we proceed in this way? There is no simple way to say this. That will depend on the nature of signal. So we should not proceed this way. Since just using the first and second order differences suffice to reduce the number of possible inputs drastically to a number easy to tackle, we would do better to use these two difference arrays. Now the minimum number of possible signal we can get is two, because however high be the order of the output we use there are always two possibilities for the second sample. Cause there is no such higher order for the second sample. If we provide the first two samples and there is no zero in the first order difference then this way we can reach to the exact input we have used. Otherwise we will get a collection of possible input signals, which are tolerant to each other.

The disadvantages in this process:

1. Firstly, the maximum error in the difference array is the step used in the preset value. But in reconstruction the error increases successively.
2. Secondly for higher dimensional signal there is more than one direction. Then it is difficult to pick up which direction we should follow.

To overcome the second problem we need to modify the process. Here we will not consider the consecutive sample difference. We consider the maximum and minimum differences for any sample with its adjacent neighbors.

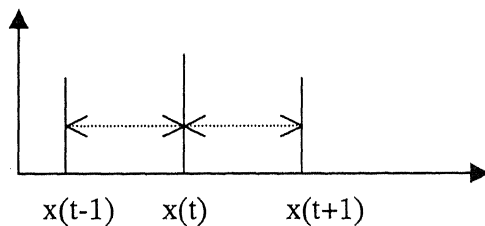


Fig.2.6 Adjacent neighbors.

So for each signal we will get two output signals one is maximal signal and another is minimal signal. If both the differences for each sample are lower than the preset value then maximal output for that sample will be one, otherwise zero and if both differences are greater

than the preset value then the output for the minimal one is zero otherwise one. That way we will get two output signals. And again applying the same idea for calculating the differences we will get two difference signals.

One will give the minimum differences for each sample considering both side differences and another will give the maximum differences. Now from these two difference- signals we will calculate the absolute difference signal. And from that, the reconstruction part is same. The reconstructed signal is shown below.

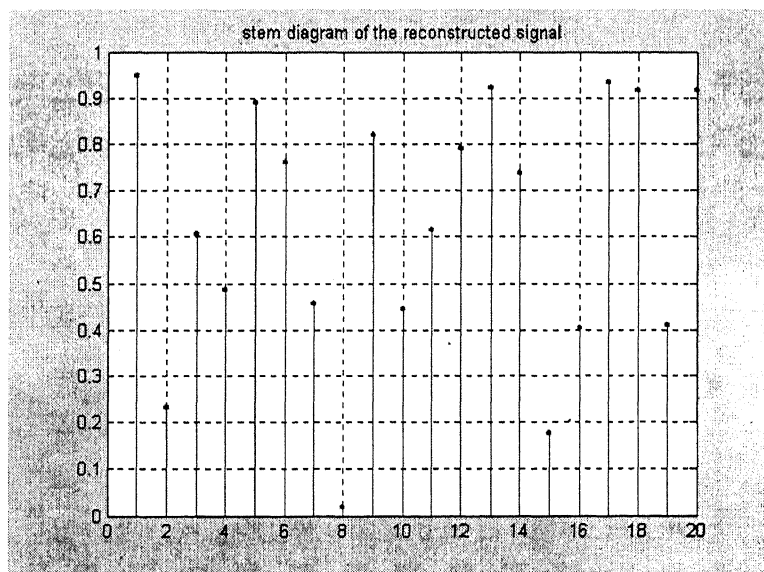
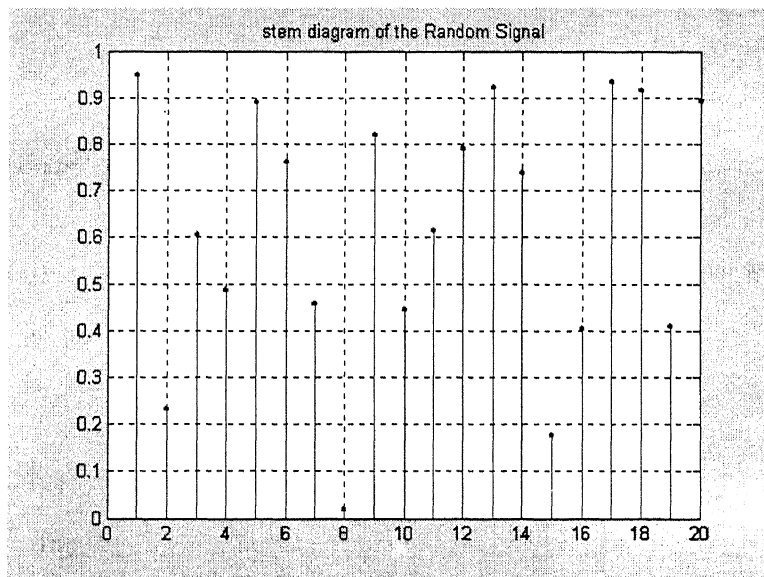


Fig.2.7 Reconstruction of 1D signals

Now we will go for images, which are two-dimensional signals. For images, we need to apply the previous one-dimensional idea in two directions, horizontal and vertical direction. So for one image we will get four output signals, two for the horizontal direction and two for the vertical. And from that the two absolute difference signals will be calculated. However, the reconstruction process is different.

Reconstruction for images

We are provided with only the first pixel value. Here the procedure is first we will calculate the first row and first column from the two absolute difference matrix. Now there is no option to check which combination of row and column will work. So we will have to check all the combination of row and columns. Now as we know the first row and column of the matrix and the first and second order difference matrices, taking each element of the first column we can calculate the possible rows and there we will shortlist the possibilities comparing the first and second order vertical absolute difference matrices. Proceeding this way if one possibility arises that row will take the place if no possibility arises then we will go for next combination of first row and column. But if more than one possibility arises we will have to proceed for both possible matrices. This way many possible inputs are generated. One thing is evident that the difference surface of an image and the complement (negative) is same. So this process will always give two images if we don't provide the more than one pixel value. The possibility of generating more than one images will arise if there is zeroes at the last row or column of the images where there is no further option to check in the forward direction. Then we need to apply the higher order differences.

Advantage of images:

1. The image pixel value lies in the range 0 to 255. So any possible image having any value beyond this limit can be removed.
2. The intensity values are integer so the round off differences will not have any error if the step is less than 0.5 and the defect of successive errors can be sorted out, and that will lead us to perfect reconstruction.

The results

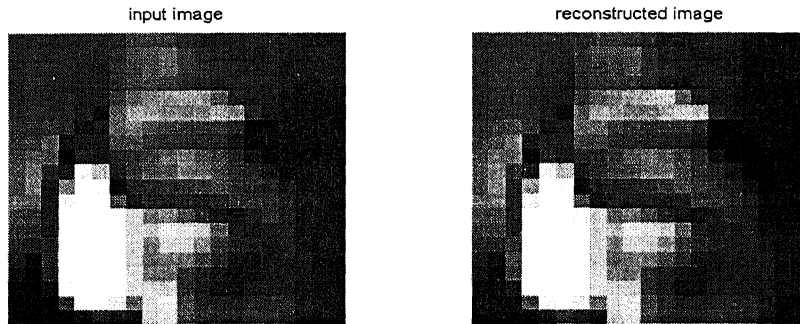


Figure 2.8. Reconstruction of 2D signals.

The left one is the input image of a “dhanesh” bird of 20x20 pixels. The right one shows the unique reconstruction when three pixel values are provided, first and second pixels of the first row and column.

Colored pictures can be treated as three components of different monochrome pictures. So this way it can be reconstructed. Other way considering the vector difference i.e. the norm of the difference vector can be used but for that case we will get only one set of difference matrices and only one image can be reconstructed. So it is not possible to reconstruct the color image.

The original “lenna” image of reduced size 50x50 pixels is used as input.

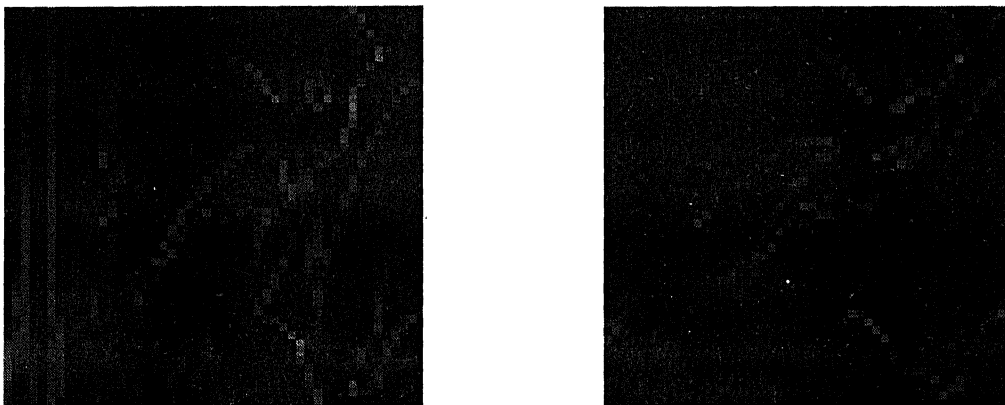
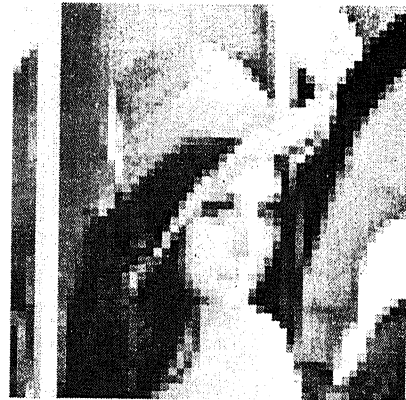


Fig.2.9. The difference images for one color component



Original Red Component



Reconstructed Red Component



Original Green Component



First Reconstructed Green Component



Second reconstructed Green Component

Fig. 2.10 (a) Reconstructed components.



Original Blue Component



First Reconstructed Blue Component



Second reconstructed Blue Component

Fig. 2.10 (b) Reconstructed components.

The reason for reconstructing two signals is that it also reconstructs the shifted negative image, which also has an identical difference structure. Now in the red component it did not generate two images, but for the rest it generated two because we are using the property of pixel values lying between 0 and 255 and providing only the first pixels intensity, a . Now the negative first i.e. $(255-a)$, will be shifted to a . So all the other pixels also will be shifted by same amount. Now if any pixel exceeds the boundary then it will be discarded else, it will be taken into account.

Considering all this outputs, using all combinations of the output components the reconstructed images are shown in the next page.



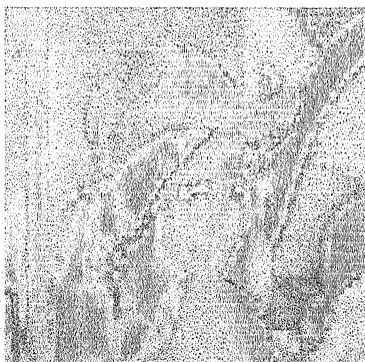
The Original Image



First Reconstructed Image



Second Reconstructed Image



Third Reconstructed Image



Fourth Reconstructed Image

Fig 2.11 Possible reconstructed images.

If any such image be found or be made deliberately such that in the difference image here are a lots of zeroes in the last part then it is possible to get more than one output images, which will look the same except a little difference in the last part where two possibilities arises. The problem of negative images can be sorted out by providing more than one pixel values along with the first one. Since the difference image is independent of the pixel intensity provided, it is possible to reconstruct the image even if any other pixel value is provided instead of the first one.

Section 3

Application of Tolerance Relations to Identify Change

In this chapter, we first try to find out what is meant by reasonable change. Next, we define the problem of change detection. After that, we break the main problem into some sub-problems and go for solving these sub-problems. All the sub-problems are broken down into the separate steps required to achieve our goal. The presentation is largely chronological: we proceed along steps that appear reasonable at first glance. As and when some problems are faced along our way, we try to eliminate them by making the necessary modifications of procedure. Most of the sub-problems have some initial steps in common: so, in our discussion of subsequent sub-problems, we sidestep those aspects that have already been treated in an earlier case, and focus instead upon new issues only. Upon solving these sub-problems, we move onto our main problem and examine how the theory of tolerance relations is applied. The results of our exercise are presented in the fifth section.

3.1 What is Reasonable Change?

There are a lot of changes that may occur through frames in a sequence of pictures captured by a camera. Of these, some, and only some, will interest us significantly, while most other changes will need to be ignored. For example, in a surveillance context, we would not like routine illumination changes (such as those that occur through the day) to trigger an alarm. Equally, in a machine vision context, changes like zooming and panning might be of no significance. We therefore require a means of intelligently ignoring what we consider to be ‘minor’ changes, without, at the same time, losing sight of even numerically slight changes that might be of interest in a specific context. Changes can be of many kinds, and what change is important is determined by the problem alone. For

example, we might be looking for changes such as movements of objects in the picture. These types of changes can be classified broadly into two categories: slow changes and rapid changes. We would perhaps like to be insensitive to the slow changes such as the movement of tree branches caused by the breeze. A reasonable change might qualify as one that involves the appearance of new object in the scene, or a shot change in a video the sequence (as against intra-shot variations).

3.2 Definition of the Problem

Originally, our object was to apply the tolerance relation for one specific application: to identify the appearance of new object in a scene where the camera is fixed. The main idea behind this approach is that when the frames of the scene are captured after a small time interval, there will be slow changes in the consecutive frames. If any new object appears in the scene, there will be a greater change, which should be detected. Now, detection of a change *per se* is not of as much importance as to determine how to neglect the small changes. This is where we are to apply the theory of tolerance. We initially need to pick up some feature of the frames that changes from frame to frame. Then, we map that feature from the physical world into the mathematical world in such a way that there is a feature sequence that will determine what change to report and what not to.

In solving the aforementioned problem, two progressively more difficult sub-problems arise that need to be solved first.

1. *First problem:* The video sequence contains several frames of different shots. Within a single shot, changes in the frame with respect to the next one are minor. However, when the shot change occurs, there will be sufficient change, that can be easily detected. Here the camera is not moved for the period of the shot.
2. *Second Problem:* Here again, we wish to detect only shot changes but the camera is now itself allowed to move. Hence the main difference with the previous case is that here (in this case) there is no fixed background.

3.3 Detection of Inter-shot Change: Fixed Camera Case.

Firstly we try with grayscale images next we will extend that idea for color images. The following way we will try to detect the change in sequences. Firstly we go to find what are the similarities between the frames and what are the dissimilarities. We need to find similarities depending upon some feature like what are the features common in both frames. So we need to identify what are the object common in both frames and what are the objects missing. To identify objects means to introduce some disjoint partition in the image so that each region in the partition is treated as a different object. After identification of the objects in both frames, we establish an object-to-object correspondence between every pair of consecutive frames of a sequence. This correspondence will help to determine the identical objects in both frames. Next job will be to make a decision of shot change depending upon the number of matches between the frames on the basis of objects and to decide how much change occurred. After that we provide a tolerance meter that will decide whether the change that occurred was enough to declare a shot change or not. So the steps are as follows:

1. Identify the objects in frames (Object Identification).
2. Build an object-to-object correspondence.
3. Find out how much changes occurred (change quantification).
4. Provide tolerance meter to take decision.

3.3.1 Object Identification.

One thing we need to keep in mind always for processing an image is the need to interpret it from a mathematical point of view. From that point of view, an object is nothing but a set of pixels of nearly same intensity or color. Whatever we are going to do in mathematical world should have a meaning in the physical world. In the physical world, an object may be of various colors. Then we will break that object into sub-objects. Whether or not one pixel is (considered to be) near to another in intensity will depend on the viewer (subjective evaluation) as well as the image itself (objective fact),

and not on either of the above alone. Therefore, we introduce a tolerance relation to determine the nearness of pixels. In grayscale images, only intensity is the deciding factor. For color images there are three color components of each frames. So each pixel has three different color intensity values. So the closeness of two colors is determined both by the color as well as the intensity.

For grayscale images:

We say one pixel is tolerant to another if the difference in intensity between the pixels, is less than some preset value of that tolerance meter.

For color images:

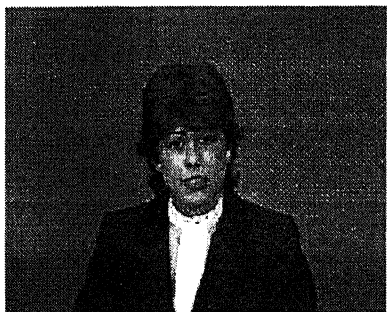
We say one pixel is tolerant to another if the difference in the max norm of the three-element color vector between the pixels, is less than some preset value of that tolerance meter.

Tolerance-preset value alpha (α):

This is the constant that is set with the tolerance meter to decide the nearness among pixels.

Now depending on this tolerance relation we try to introduce a partition in the image. To do this, we first try to break the whole set of pixels into tolerance classes. Here we are taking the whole image as a set of pixels. As we have the idea of tolerance classes, it's the class where any two elements will be tolerant to each other. We use the standard test image Claire to apply this idea so that we may first apply it to an image where there is little variation. The outputs are shown in Fig3.1 for a tolerance meter preset value of 70.

Next, in these images we set background as white and the identified object is to its original color. We see that the first object identified the dress and hair of Claire as one object. In the second object also it identifies the same but there is a little difference of including some new pixels and excluding some pixels. The same also happens to the third object.



The Original Image



The First Object

Fig 3.1(a) The original image and first object using tolerance class.

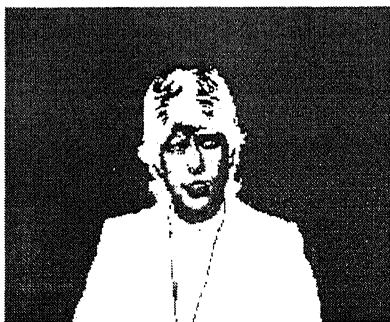


The Second Object



The Third Object

Fig.3.1 (b) First three identified objects of "Claire" using tolerance classes



The Fourth Object



The Fifth Object

Fig.3.1(c) 60th and 110th objects of "Claire" using tolerance classes.

The images, shown in the Fig3.1(c), are the sixtieth and one hundred tenth objects accordingly. The image referred as fourth object identifies the whole background and some part of her face. The fifth one recognizes the face and some part of background. Proceeding with this idea several problems arise. These are:

1. Any other pixels outside a class will be a member of that class if it is tolerant to all the member of that class. So there is a maximum and minimum limit of intensity value. Any pixel intensity just above the maximum value will not be tolerant to the minimum one. So another tolerant class will arise including the outsider and excluding the extreme one. That way there will be huge overlapping among tolerance classes. There may be difference of only one pixel between classes.

Let's make it clear with an example. Let us consider one image having pixel intensity values all integers 0 to 255. So if the tolerance-preset value is 5 then first class will contain pixels whose intensities are in the range 0 to 5. And the next class will contain the pixels ranging 1 to 6. So the overlapping pixels intensities will be 1 to 5. So one object will be selected several times or its parts may be present with other objects. But our objective is to pick up disjoint objects.

To overcome this problem, we need to reduce the number of classes. Here this overlapping is inevitable because tolerance relation is not transitive. So it is not possible to get a disjoint partition. There are a lot of redundant classes. To discard redundant classes we pick up only the so-called basis classes. With the basis classes the outputs are shown in Fig3.2 with tolerance meter preset $\alpha = 70$.

In these images we look for the value of tolerance meter preset value 70 there are four basis classes. First basis class identifies her dresses and hairs as one object. Next basis class identifies the whole background and some part of her face. Here one important thing to notice is that these two are identified as two different patches. In the next two classes contains her face and the tie. The third basis class detects that part of her face what is brightened because of light. Although the brightened part is not brightened drastically compared to the neighbors still it has been detected as different class. The

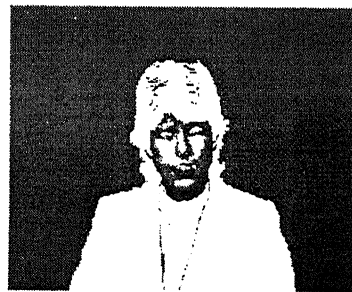
fourth object her tie is sudden bright compared to the neighbors. So it can be accepted as a different object.



The Original Image

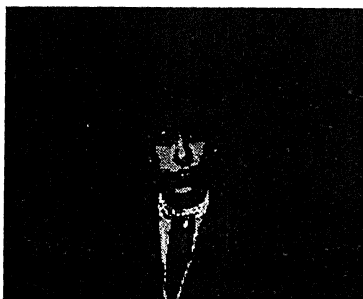


The First Object

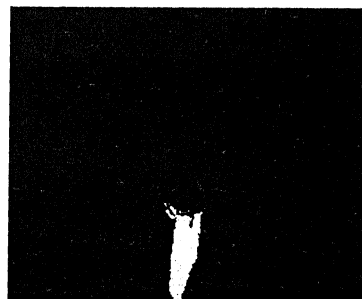


The Second Object

Fig.3.2 (a) Identified object using tolerance bases.



The Third Object



The Fourth Object

Fig 3.2(b) Third and fourth objects using tolerance bases.

Here one thing we need to mention is that basis classes of tolerance relation are not unique. So another basis class may give better result. But the problem of dividing the face into two classes will not be removed. Therefore, we did not consider neglecting the slow variation between neighbor pixels.

2. As it is shown in the results, after using the idea of tolerance classes we are getting pixels scattered in the image what cannot be taken as an object, that is to say that these pixels are not spatially neighbors. So the outcome is some arbitrary patch in the image. Thus whole classes cannot conveniently be treated as an object. An object should be spatially close as well as close in intensity.

Even if we use tolerance basis classes this problem is not removed. Keeping this in mind for identification of objects we need to modify the definition of tolerance relation what we have made to define the nearness of pixels.

For grayscale images:

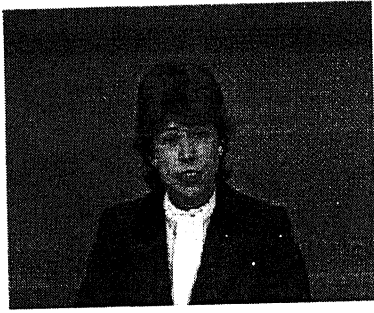
We say one pixel is tolerant to the other if the difference in intensity between the pixels, is less than some preset value of that tolerance meter and also spatially a neighbor.

For color images:

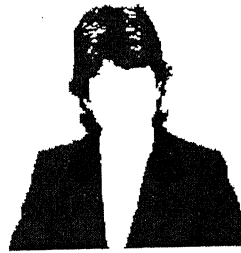
We say one pixel is tolerant to the other if the difference in the max norm of the three element color vector between the pixels is less than some preset value of that tolerance meter and also spatially a neighbor.

Now with this modification, the outcomes are shown in Fig3.3.

Here there are a lot of objects with very small number of pixels what we did not show. The objects of the previous tolerance basis are now broken in such a way that each patch is considered as new object. Still it has broken the face into two classes.



The Original Image



The First Object

Fig 3.3(a) Original and First object using Modification.

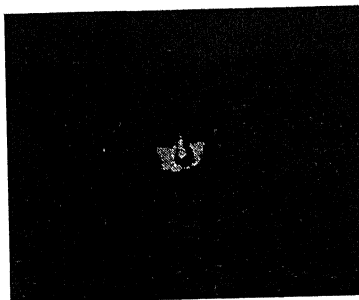


The Second Object



The Third Object

Fig 3.3(b) Identified objects using Modification.



The Fourth Object



The Fifth Object

Fig 3.3(c) Identified objects using Modification.

So we can see the object identifications is not complete. The problem arises because of 'slow' variations. Lets consider a ball as an object. Here the ball has two intensity regions. One part is bright side and another part is the dark side where light is not falling directly. And the transition from one region to another region is not abrupt. There is gradual change of intensity. Tolerance class cannot neglect gradual changes. So the way we are progressing will not detect such an object completely.

To overcome this trouble we need to go for transitive closure. It will pick up the set where gradual change will be neglected. Transitive closure can be thought as imposing transitivity on our previous relation. Since transitivity is imposed we will get a disjoint partition in the image. To get desired partition we need to first set the value of alpha, the tolerance meter preset value, by trial and error method. After setting suitable value of $\alpha = 4$ the results are coming like the following:

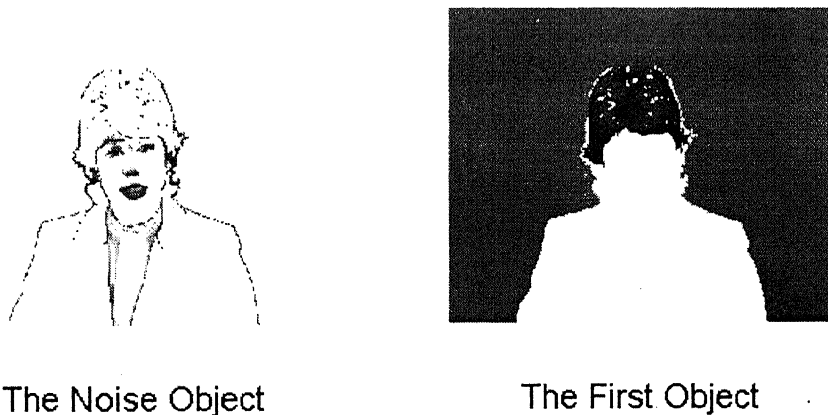


Fig.3.4 (a) Original and First object using Transitive closure.

Here we got five major objects. First one as shown is not really an object. It is the tiny objects collected together. The second one is the first major object. It identifies the whole background and her hair. It is because in this image there is a smoothness of the intensity of hair with the background at some region. The third object identifies her whole face as another object. The fourth and fifth one detects her dress as different object, since they are not connected. And the eighth object identifies her Tie. Total ten objects are identified. We have only shown the major objects.



The Third Object



The Fifth Object



The Sixth Object



The Eighth Object

Fig.3.4 (b) Objects identified using transitive closure.

Now the major objects are identified, as we wanted it to be. But still there some minor obstacles in this way. These are

- 1 There are a lot of small objects containing one or very small number of pixels. These are not any objects in the picture. It is of no harm if we consider them but to reduce computational complexity we neglect this tiny objects and call as noise in the picture or speckles. Or if it is of interest, these can be accepted as objects.

To get out of this trouble we provide one threshold value lets say it as β (beta), as we had set it for the previous result to 35, what is nothing but the minimum limit of the size of objects. If the size of any object is less than this threshold value it will be

discarded as noise other wise it will be accepted. So this threshold will act as a tuning parameter of the tolerance meter.

- 2 Another trouble will appear depending upon the images. If any image contains any object with various color or abrupt intensity, then that object will not be detected as a single object. The uniform parts of the objects will be selected as different objects. Again the uniformity will depend on the value of alpha, the tolerance meter preset value. If the value is not set properly this small sub-objects will be discarded as noise. Thus the whole object may remain unidentified. The worst case of this will happen if the object is covered with some textures.

A better way to overcome this barrier is to set the value of alpha, the tolerance meter preset value, and beta, the object threshold value, very carefully. But if the object is having a texture, then this will not work. In section 4, we have discussed how to identify textures. Now after identifying them as sub-objects, one decision can be made comparing with other frames. That is if these sub-objects remain spatially close, then all of them can be treated as one object. But this decision cannot be made depending upon only one frame. If we try to identify the small sub-objects as one single object, by increasing the value of the tolerance meter preset value alpha until it exceeds the maximum difference between these sub-objects, i.e. until it identifies the whole object, the performance of the texture segmentation could be impaired severely in other regions of the image. It can be accepted only if it does not merge two different objects into one.

3.3.2 Building an Object-to-Object Correspondence Between Frames.

Here we try to determine the object-to-object correspondence between the consecutive frames. If the frames are of the same shot, most of the objects of the previous frames can be found in the next frame also. So we need a means what will match one particular object in one frame and another particular object in the other frame. So we need to get this one to one mapping.

Observing the properties that objects (ought to) preserve from frame to frame can establish this correspondence. These properties may include the size, intensity and location among other properties. From frame to frame, one may expect very slow changes of size. The same thing also holds for intensity.

For gray scale images, we define a distance measure. Next for a particular object we pick up its closest approximation on another set. Here we consider frames as a set of objects. And we are trying to find a one-to-one mapping between two frames.

Next considering the intensity we look for the closest approximation of the identified objects. We give intensity mapping the first priority because the size may vary within a limited range but the intensity is more or less remains same. There may be problem like two objects may be mapped to a single object. If this problem arises in the intensity mapping, then we check for the size mapping. Most of the cases this problem disappear if we consider this two mapping simultaneously. Again if for any object there is two images in the range frame, then also first we check for intensity mapping and then we check for the size mapping.

Again there may be the case that both mapping shows one-to-two or two-to-one mapping. This problem arises because of the merging of two objects into one or splitting of one object into two. This can be taken into consideration if we check for the size of one is nearly equal to the total size of the two. Then we take this multiple object into one object. If this also cannot make any decision which one is the exact one then we can go for the position. How much position is shifted can serve as a distance measure. That way we can pick up the closest one.

All these possibilities may occur when there are exactly two same objects in one frame and one in the other frame. In that case the aforementioned (position) shifting measure will work to distinguish/identify objects.

In any case, we do not resort to the shifting measure unless we have any movement. For this correspondence we define the following constants:

- 1 *The intensity constant γ (gamma):* This is one threshold value of the tolerance meter provided for closeness in intensity mapping.
- 2 *The size constant δ (delta):* This is the threshold value for the tolerance provided for determining the closeness in size mapping.
- 3 *The shifting constant φ (phi):* This constant will act as the tolerance meter threshold provided for determining the closeness in the shifting mapping.

The value of this tolerance meter preset value should be chosen in such a way that major objects (objects with large size) would get priority. In most cases of more than one pre-image or image, there is a large difference in the closeness of intensity mapping and size mapping. There these constants should be chosen in such away that the closest one would be picked up, also the size should be kept in mind.

With all this preset value the block diagram is shown in Fig3.5 to find out the correspondence.

The Block Diagram

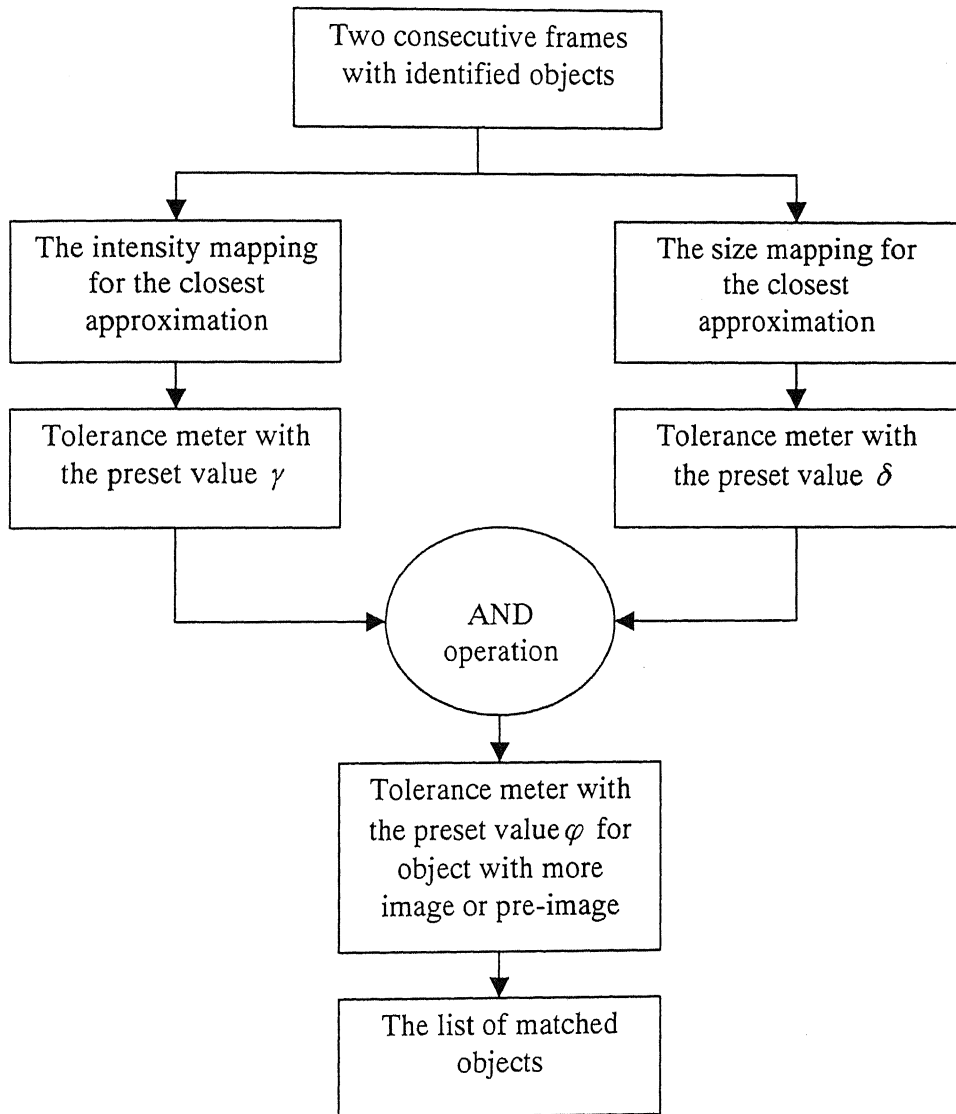
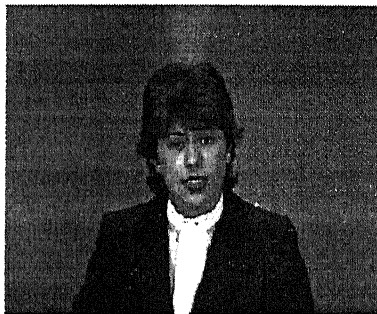


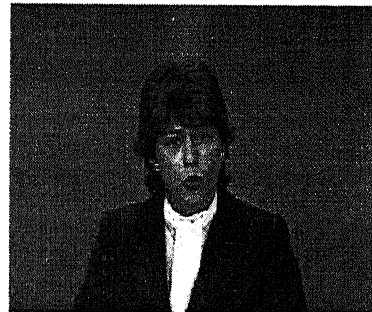
Fig. 3.5 Block diagram to determine object-object correspondence

For color images instead of intensity mapping, we go for color mapping. There first for each object color vector is determined. Then the Euclidian distance between color vectors are taken. To determine the color vector, we pick up each of red, green and blue monochrome image and take the mean of that. So the output will be a vector of three elements. Here the max norm also can be chosen. But for matching, to incorporate high accuracy we chose Euclidian distance.

The following result will clarify this idea.



The First Frame



The Fourth frame

Fig.3.6 Frames used to establish object-to-object correspondence



The Noise Object



The First Object

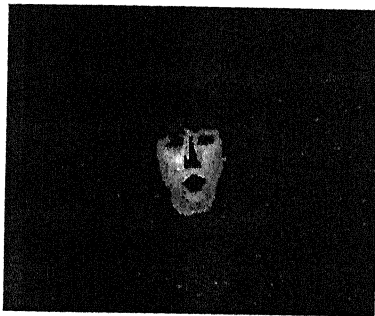
Fig 3.7(a) Objects of the fourth frame.

For the first frame the extracted objects are shown earlier pages. For the fourth frame total fourteen objects are identified. Out of which we have shown only five major

objects. So between these two frames first one has eleven major objects and second one has fourteen major objects.

The Procedure with Results

1. Step1: Map both the sets of objects to mathematical domain considering the intensity.
2. Step2: Map both the set of objects to mathematical domain considering the size.
3. Step3: For the intensity mapping, find the close approximation from one set of objects to other set of objects both ways.



The Second Object



The Sixth Object



The Seventh Object



The Twelfth Object

Fig.3.7 (b) Objects of the fourth frame.

4. Step4: Find the close approximation from one set of objects to other set of objects both way considering the size mapping.

5. Step5: Find the mapping which are common for intensity mapping and size mapping.
6. Step6: Pick up all one to one correspondence both ways.
7. Step7: For the mappings, which are many to one, find the closest approximation considering size and intensity mapping. If that also fails to reduce it into one to one mapping, then pick up the closest one considering the shifting of object. This can be done taking the inter section of objects.

For these two frames the intensity array is

Pr12=[121, 140, 145, 44, 43, 45, 197, 234, 213, 192] for the set of ten objects.

Pr22=[120, 143, 97, 175, 42, 43, 44, 90, 97, 200, 232, 234, 190, 191] for the set of 14 objects.

Considering the size mapping

Pr11=[75152, 116, 3220, 36, 9342, 7710, 66, 1940, 31, 141] for the set of ten objects.

Pr21=[75075, 3252, 39, 63, 41, 9321, 7760, 41, 52, 77, 47, 1903, 176, 41] for the set of 14 objects.

The distance matrix is shown below foe both intensity mapping and size mapping.

Test1=	1	22	24	54	79	78	77	31	24	79	111	113	69	70
	20	3	43	35	98	97	96	50	43	60	92	94	50	51
	25	2	48	30	103	102	101	55	48	55	87	89	45	46
	76	99	53	131	2	1	0	46	53	156	188	190	146	147
	77	100	54	132	1	0	1	47	54	157	189	191	147	148
	75	98	52	130	3	2	1	45	52	155	187	189	145	146
	77	54	100	22	155	154	153	107	100	3	35	37	7	6
	114	91	137	59	192	191	190	144	137	34	2	0	44	43
	93	70	116	38	171	170	169	123	116	13	19	21	23	22
	72	49	95	17	150	149	148	102	95	8	40	42	2	1

Test2=

77	71900	75113	75089	75111	65831	67392	75111	75100	75075	75105	73249	74976	75111
74959	3136	77	53	75	9205	7644	75	64	39	69	1787	60	75
71855	32	3181	3157	3179	6101	4540	3179	3168	3143	3173	1317	3044	3179
75039	3216	3	27	5	9285	7724	5	16	41	11	1867	140	5
65733	6090	9303	9279	9301	21	1582	9301	9290	9265	9295	7439	9166	9301
67365	4458	7671	7647	7669	1611	50	7669	7658	7633	7663	5807	7534	7669
75009	3186	27	3	25	9255	7694	25	14	11	19	1837	110	25
73135	1312	1901	1877	1899	7381	5820	1899	1888	1863	1863	37	1764	1899
75036	3213	0	24	2	9282	7721	2	13	38	38	1864	137	2
74934	3111	102	78	100	9180	7619	100	89	64	64	1762	35	100

Considering the values we set the value of γ to 10 and the value of δ to 90. The numbers in the following result are the object numbers.

For intensity mapping, the mapping that passed the tolerance meter is:

1	2	3	4	5	6	4	5	6	4	5	6	7	10	8	8	7	10	7	10
1	2	2	5	5	5	6	6	6	7	7	7	10	10	11	12	13	13	14	14

For the Size mapping the outputs of tolerance meter are:

1	3	2	4	7	9	2	4	7	9	10	2	4	7	9	5	6	2	4	7	9
1	2	3	3	3	3	4	4	4	4	4	5	5	5	5	6	7	8	8	8	8

2	4	7	9	10	2	4	7	9	10	2	4	7	9	8	2	10	2	4	7	9
9	9	9	9	9	10	10	10	10	10	11	11	11	11	12	13	13	14	14	14	14

Now the mappings common to both of this mapping are:

1	3	4	5	6	7	10	8	10	7
1	2	5	6	7	10	10	12	13	14

Now Out of this the one to one mappings are:

$$\begin{bmatrix} 1 & 3 & 4 & 5 & 6 & 8 \\ 1 & 2 & 5 & 6 & 7 & 12 \end{bmatrix}$$

For the rest, we go for closest approximation, first considering the intensity mapping then the size mapping. That reduces these to the following one to one mapping

$$\begin{bmatrix} 7 & 10 \\ 10 & 13 \end{bmatrix}$$

So the correspondence table is

Objects From the First Frame	Objects From the Fourth Frame
First Object	First Object
Third Object	Second Object
Fourth Object	Fifth Object
Fifth Object	Sixth Object
Sixth Object	Seventh Object
Seventh Object	Tenth Object
Eighth Object	Twelfth Object
Tenth Object	Thirteenth Object

Table 3.1. Correspondence between the first and the fourth frame.

3.3.3 Determination of The Amount of Change that has Occurred.

Based on the established correspondence as discussed in the previous sub-section we make a list of matched objects. First we thought if most of the objects were found in the next frame we conclude there would be no shot change. There would be a tolerance meter with some preset value what will represent the minimum number of object matches required to conclude no change.

But this causes one problem. There are many objects with small size. These small objects also achieve the same weight with the major objects in making decision. So where these small objects dominate the major objects, the results may become erroneous. So we need to give more weight to the major objects. Neglecting the minor objects will give an acceptable answer but that will not be accurate enough.

So we consider the total size of the matched objects. That will give the amount of the frame image matched with the next frame. This also gives the measure of the change that has occurred.

For the previous correspondence, we present the computation of the amount of change:

The amount of area in terms of number of pixels of the total matched objects is 97607. Total area of the image is 101376. So unmatched area is 3769.

So the change index in percentage is 3.7%.

3.3.4 Tolerance Meter to Take Decision.

The last tolerance meter will be provided to make decision depending on the change occurred or in other way on the similarities from frame-to-frame.

The matching constant μ (μ): This constant will act as the tolerance meter preset value, provided for determining the matching of frames with next frames.

This constant is the minimum value of the ratio of size of the matched part of image with the dissimilar part of the image. So if the matched part generates the ratio more than this preset value there will be no change, otherwise it will detect as shot change.

For the previous correspondence we set the value of $\mu=15\%$.

So the value of occurred $\approx 3.7\%$. So it passes the tolerance meter. So there is no shot change.

3.4 Detection of Inter-shot Change: Moving Camera Case.

With all these four steps if a sequence is passed, we found that a shot change is detected satisfactorily. The last step was the decision. This decision-making can be modified suitably along with the change detection step to determine whether a new object appeared or not. Only the tolerance meter preset values should be chosen properly with intense care. For the problem of shot change detection when camera is moving the steps of the previous problem are also applicable here. The basic difference with the previous problem is that there, the background remained fixed, so that change consisted only of object movement. Here since the camera is moving, the amount of change will large. Experiments show that in general, the value of the preset value for establishing correspondence should be chosen properly, i.e. more relaxed than for the previous case.

Also the preset value of the last (μ) tolerance meter to detect shot change should be greater than the previous problem.

3.5 Detection of the Appearance of New Objects In a Sequence

To detect the appearance of a specific new object in a largely stationary scene, we need to have some knowledge of the object of interest. It is not the case that the object will appear all of a sudden in the scene. The appearance of an object will be gradual. So there

is a basic difference with the previous problem. If we know the size of our sought object we will consider all the objects below that value as noise. We will go to detect the objects larger than that value. So when an object will start to appear until it appears to that extent tolerance meter will not detect that. Whenever the object exceeds the limiting value it will be counted as an object. So during the transition period, that object will not be having any pre-image when we attempt a correspondence mapping.

So if we count the number of objects at each frame, whenever the number increases the deciding tolerance meter will detect the appearance. This way it is also possible to detect the disappearance of object from the scene. But if any major object starts disappearing with the appearance of the new major object in such a way that at the transition period there is no change in the total number of objects then it is not possible to detect. Although this may only rarely happen, we cannot skip the step of establishing the correspondence from frame to frame. The deciding step will be completely different. Here we not only care for how much change occurred but also the major objects are mapped or not. If not which frame it belongs to? If it belongs to the previous frame then object disappearance will be declared and if it belongs to the present frame then appearance of new object will be declared. So for this problem the steps are:

- 1 Step 1: Identification of objects choosing a suitable identification preset value.
- 2 Step 2: Establish object-to-object correspondence.
- 3 Step 3: Find the amount of change that has occurred, the number of major objects with a one-to-one map, the number of major objects in the frames.
- 4 Step 4: Take a decision tolerance meter to detect same shot and appearance of new object.

In this problem we cannot disregard the problem of shot change detection. Because if shot change occurs the objects of the previous frame will be mapped to the closest approximation and there new objects are may not be the one of our interest.

Section 4

Texture Identification

In this section we discussed how tolerance relation could be applied for identification of objects containing textures. We showed segmentation of one test image containing textures.

4.1 Quantifying Differences in Texture.

Let us consider the image shown below. In this image there are several objects. From our point of view, the floor and the wall are also objects. In all, there are two sofa sets. One sofa is covered with a cover not having uniform color. It is like a spatial pattern.

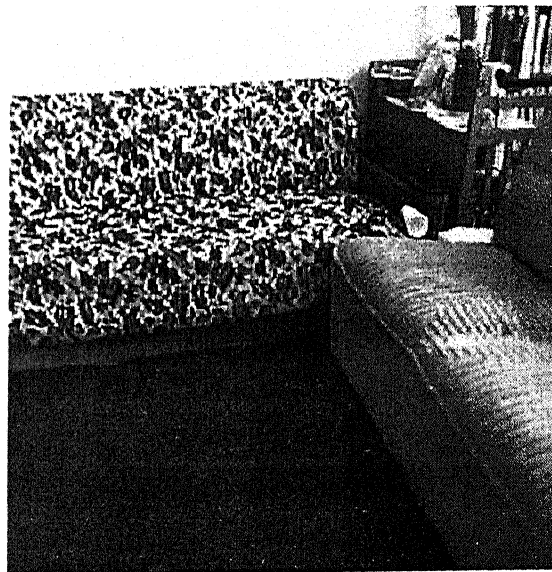


Fig. 4.1 Example of textural objects.

feature will be extracted block-wise but the classification will be made pixel-wise. That is, for each pixel we consider a block around it and extract the feature and move to the next pixel and the tolerance meter will compare these features that are pixel-wise.

There are two classes of approaches to segment the images containing textures. The first approach is to extract the textural features with statistical tools. As suggested by Haralick *et al.* the fourteen statistical features are:

Notation used:

$p(i, j)$: (i, j) th entry in a *normalized* gray-tone spatial-dependence matrix,
 $= P(i, j) / \text{mean}$

$p_x(i)$: i th entry in the marginal probability matrix obtained by summing the rows of $p(i, i)$, $= \sum_{j=1}^{N_g} p(i, j)$ where N_g = number of distinct gray level in the quantized image.

$$p_y(j) = \sum_{i=1}^{N_g} p(i, j).$$

$$p_{x+y}(k) = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i, j), \quad k = 2, 3, \dots, 2N_g$$

$$i+j=k$$

$$p_{x-y}(k) = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i, j), \quad k = 0, 1, \dots, N_g - 1.$$

$$|i-j|=k$$

HX and HY are entropies of p_x and p_y . $HXY = f_9$.

$$HXY1 = - \sum_i \sum_j p(i, j) \log \{p_x(i) p_y(j)\}.$$

$$HXY2 = - \sum_i \sum_j p_x(i) p_y(j) \log \{p_x(i) p_y(j)\}.$$

$$Q(i, j) = \sum_k \frac{p(i, k)p(k, j)}{p_x(i)p_y(j)}.$$

Textural Features:

1. Angular second moment: $f_1 = \sum_i \sum_j \{p(i, j)\}^2.$

2. Contrast:
$$f_2 = \sum_{n=0}^{N_g-1} n^2 \left\{ \sum_{i=1}^{N_g} \sum_{\substack{j=1 \\ |i-j|=n}}^{N_g} p(i, j) \right\}$$

3. Correlation:
$$f_3 = \frac{\sum_i \sum_j (ij) p(i, j) - \mu_x \mu_y}{\sigma_x \sigma_y}$$

4. Sum of squares:
$$f_4 = \sum_i \sum_j (i - \mu)^2 p(i, j).$$

5. Inverse Difference Moment:
$$f_5 = \sum_i \sum_j \frac{1}{1 + (i - j)^2} p(i, j).$$

6. Sum Average:
$$f_6 = \sum_{i=2}^{2N_g} i p_{x+y}(i).$$

7. Sum Variance:
$$f_7 = \sum_{i=2}^{2N_g} (i - f_6)^2 p_{x+y}(i).$$

8. Sum Entropy:
$$f_8 = - \sum_{i=2}^{2N_g} p_{x+y}(i) \log\{p_{x+y}(i)\}.$$

9. Entropy:
$$f_9 = - \sum_i \sum_j p(i, j) \log(p(i, j)).$$

10. Difference Variance: $f_{10} = \text{variance of } p_{x-y}.$

11. Difference entropy:
$$f_{11} = - \sum_{i=0}^{N_g-1} p_{x-y}(i) \log\{p_{x-y}(i)\}.$$

12. Information Measures of correlation:
$$f_{12} = \frac{HXY - HXY1}{\max\{HX, HY\}}$$

13. Information Measures of correlation:
$$f_{13} = (1 - \exp[-2.0(HXY2 - HXY)])^{1/2}.$$

14. Maximal correlation coefficient. $f_{14} = (\text{Second Largest Eigen Value of } Q)^{1/2}.$

To apply these to images we first need to identify four P matrices in four directions - horizontal, vertical, left diagonal, right diagonal. These matrices are created as described by Haralick *et al.* The number $P(i, j)$ is how many times the gray tone level i and j have been neighbors in the original image along the particular direction. The neighborhood is also determined by a factor d , which is the distance of pixels in a particular direction.

Another approach is to carry out a frequency analysis, such as by finding the DCT coefficients, or by Gabor filtering. Other approaches are structural approaches, using co-occurrence matrices.

Anyway, we choose a block of sixteen by sixteen for the standard test texture image. Then we will try to extract a feature such as frequency variation; that makes a block similar or dissimilar to the neighboring blocks. We take a two dimensional discrete cosine transform and form a vector of three elements of low, medium and high frequency components. If the blocks are taken from different textures, then at least in one of the components, there will be a large enough difference to keep those blocks apart.

For an eight by eight block, the first four elements will be treated as low frequency components. Last nine components will be treated as high frequency components. And the rest will be treated as middle frequency components. This frequency partitioning can be varied to achieve smoothness.

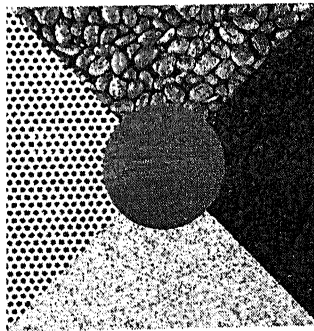


Fig.4.2 Test image used for texture segmentation.

From dct2 block processing, with a block size of 16x16, the low frequency part is shown in Fig 4.3. This is the image of only first component of the low frequency part. Other frequency parts, middle and high frequency, were found unsuitable for segmentation because they could not bring uniformity in the image. Taking the whole of the low frequency part as shown in fig 4.3 also distorts the uniformity. Only the first element of the low frequency part (Fig 4.4,) is more suitable. Here, all other middle and higher frequency variations are not giving good result. To find out which frequency part will give good result is a trial and error process. We first divide the whole dct2 output into three frequency parts. If this is not enough to give segment-to-segment distinction we will go for more sub-division like horizontal/vertical low frequency, horizontal/vertical high frequency and so on. Here the idea is that if one region is different from the other at all, then it must manifest as a distinction in one or other of the frequency components. So first we segmented this image and then took the partition in the original. To segment this image we should not use transitive closure as this will lead to erroneous results. Here tolerance basis will work well. Here for several tolerance meter preset value we will have to decide. That is we use a parallel combination of some of the tolerance meters. The number will depend upon the range of values of the dct2 output matrix.

The maximum value of the image Fig.4.4 is 3556 and the minimum value is 363. For different tolerance meter preset values the outputs are shown in the next page. The tolerance meter with preset 2701 classifies the whole image into two classes as shown in Fig.4.5. Now Fig 4.6 shows the result of channel2 tolerance meter with preset 1890. This also classifies into two classes.

Next the output of the third channel is shown in Fig 4.8 with the preset value 850. Here we get four classes. Now out of these three channel outputs we pick up the connected objects.

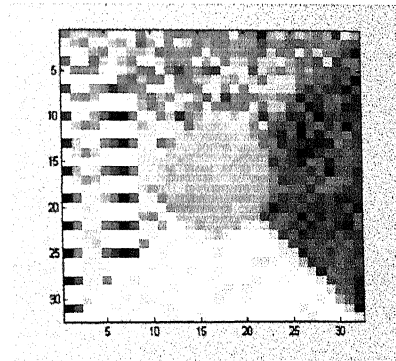


Fig. 4.3 The whole Low frequency Part.

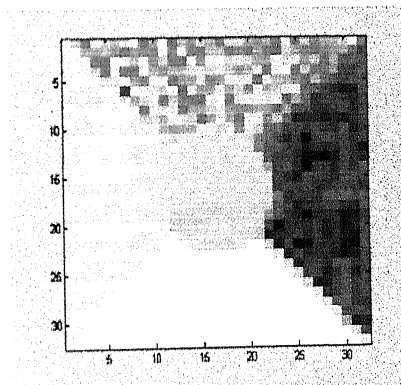


Fig.4.4. Only the first element of the low freq. Part.

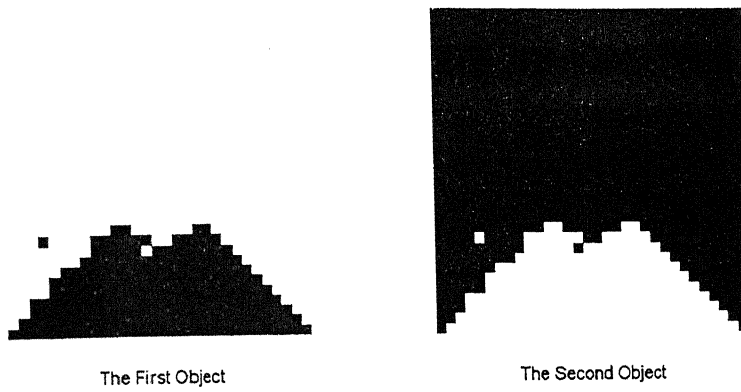


Fig.4.5 Partitions with preset 2701.

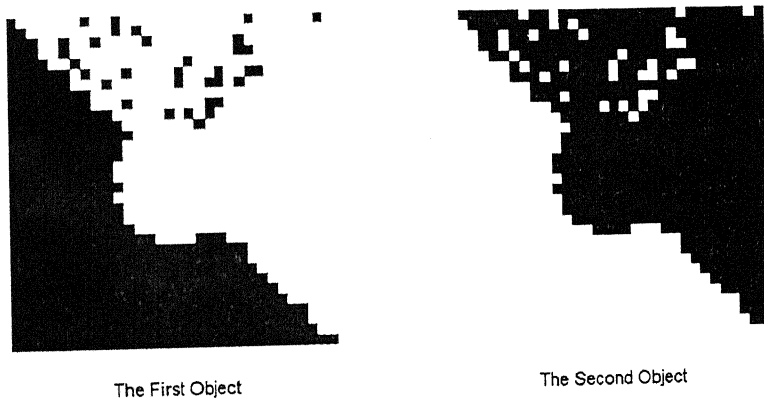


Fig.4.6 Partitions with preset 1890.

From these two channel outputs the introduced partitions are shown.

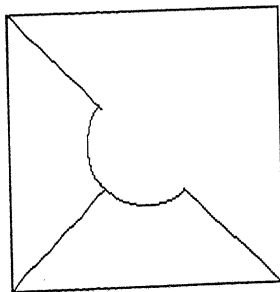
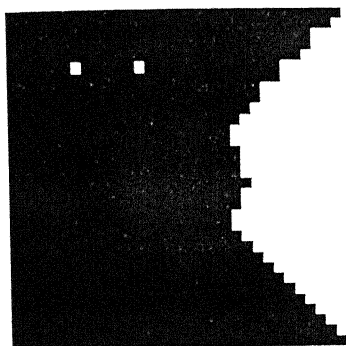
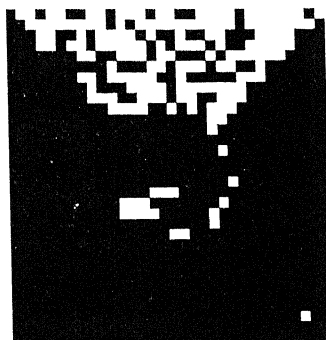


Fig.4.7 Partitions induced with the results of first and second channels.

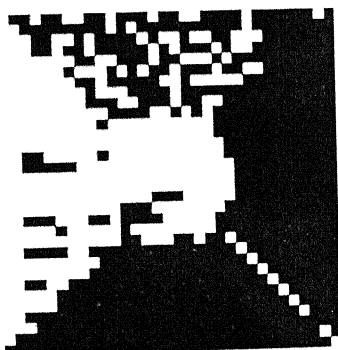


The First Object

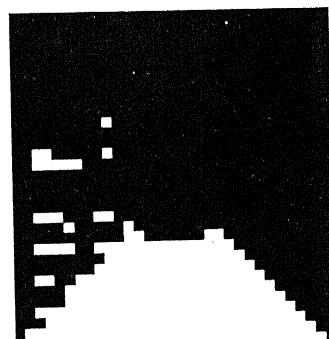


The Second Object

Fig.4.8 (a) first two partitions with preset 850.

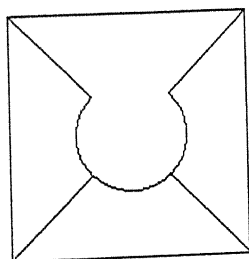


The Third Object

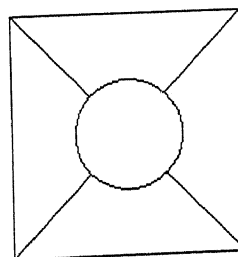


The Fourth Object

Fig.4.8 (b) Third and fourth partitions with preset 850.



(a)



(b)

Fig.4.9 Partitions induced with all tolerance meters' outputs considered.

For these three channels outputs the induced partitions is shown in the previous page in Fig 4.9. Fig 4.9(a) shows the partition when the first object of the third channel is included. So the final identified partitions for each channel are shown in Fig 4.10 after smoothing the objects.

Now from this database we calculated the objects larger than some specified size. That is from these three sets of partitioning we pick up objects by taking their intersection.

The identified objects are shown in Fig.4.11. After that we have taken back these partitions in our original test image. Fig 4.12 shows the identified objects of the test image.

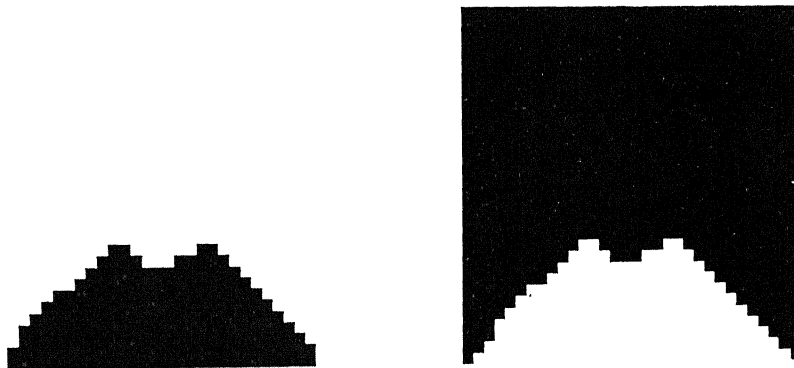


Fig 4.10(a) Identified Partitions after smoothing for channel 1.

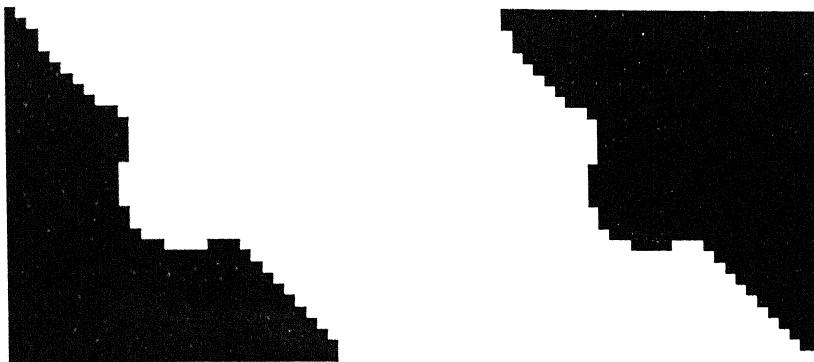


Fig 4.10(b) Identified Partitions after smoothing for channel 2.

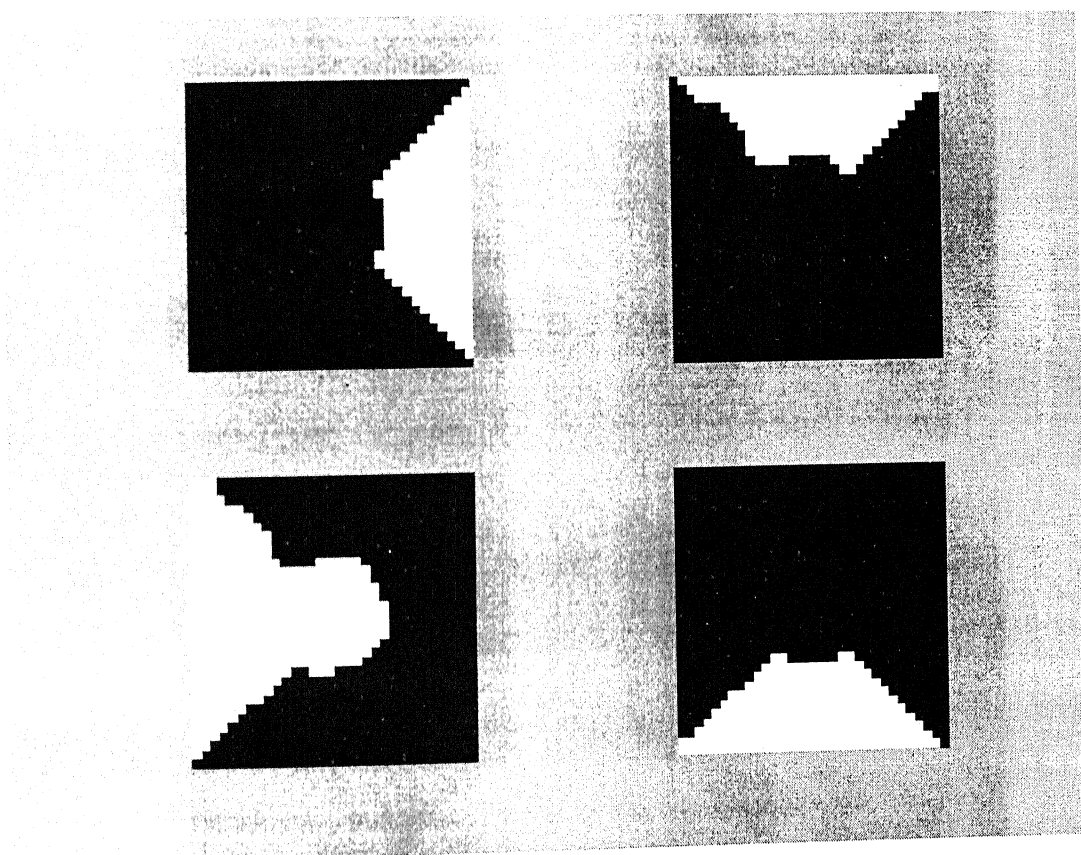
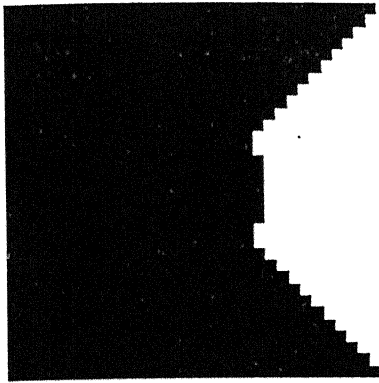
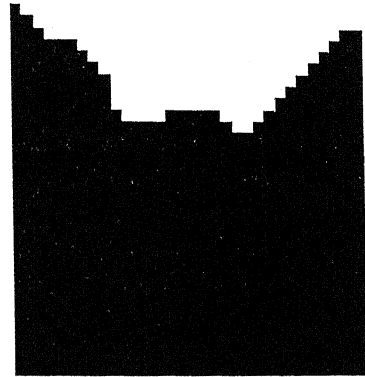


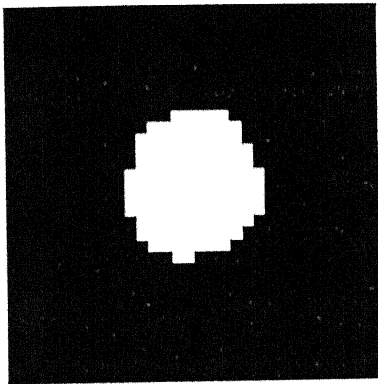
Fig 4.10(c) Identified Partitions after smoothing.



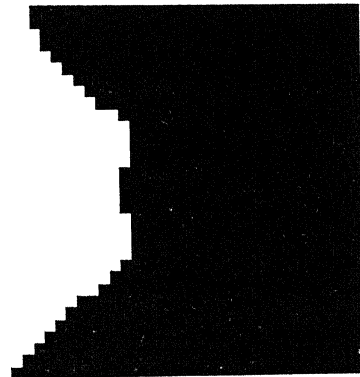
The First Object



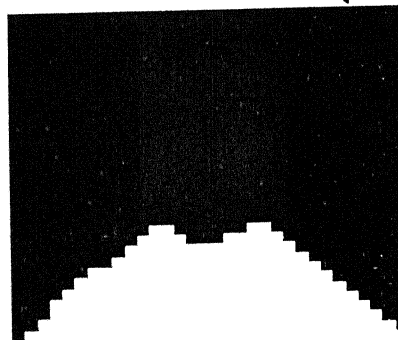
The Second Object



The Third Object

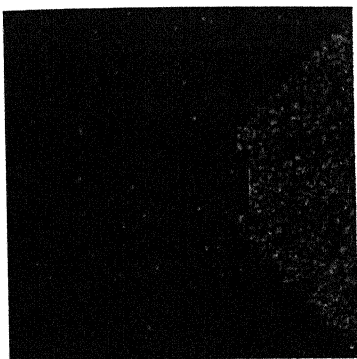


The Fourth Object

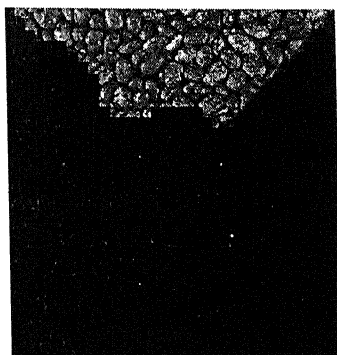


The Fifth Object

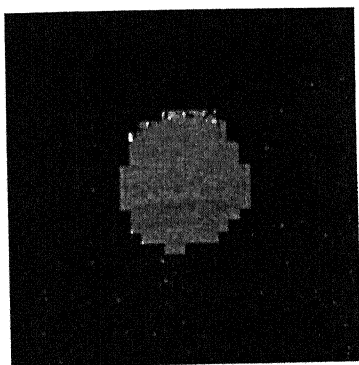
Fig 4.11 Identified objects.



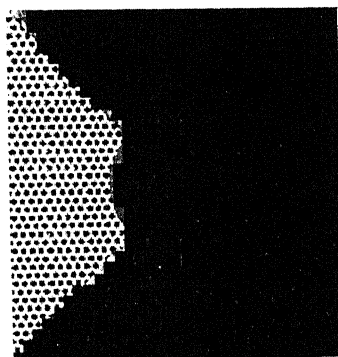
The First Object



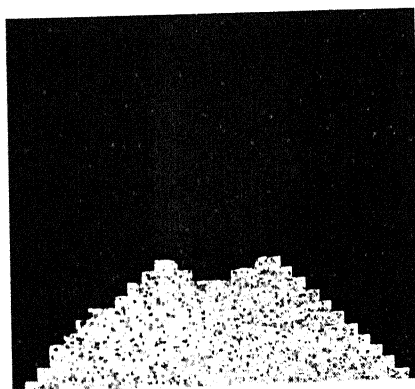
The Second Object



The Third Object



The Fourth Object



The Fifth Object

Fig 4.12. Identified objects in the original test image

There are also other advanced tools to segment texture. Once the mapping is complete, and then we can use tolerance relation to identify objects. In this method, we had to provide three tolerance meter channels to introduce more partitions. And also smoothing program incorporates error in it. So for a video sequence containing textures we just need to set the tolerance meter preset value for the first frame. Then the method proposed in the last section can be used to detect reasonable changes of interest.

Section 5

Algorithms and Results

Algorithm Proposed for the Detection of Reasonable Change

The notations used here:

α = The preset for the tolerance to determine closeness between pixels.

β = The threshold giving the minimum size of objects.

γ = The preset for the tolerance to determine closeness in intensity mapping.

δ = The preset for the tolerance to determine closeness in size mapping.

φ = The preset for the tolerance to determine closeness in the sense of overlapped region.

Step 1: First we set the value of α and β for an image from the sequence.

For an image picked up from the sequence, for any arbitrary value of β , with various α the outcome is shown in tabular form in the next page. Here we see for low value of α , the number of objects is much larger.

As α is increasing, the number of objects is decreasing. We chose the optimum value of $\alpha = 15$. For $\alpha > 15$ the number of objects is less, but it is merging two objects into one. For the value 14 and 15 it is giving same number of objects but for 15 the size of the noise object is smaller. For this value $\alpha = 15$, we show the result for various β in tabular form. Here we will care for the major objects only. Now we see that for $50 > \beta > 35$ total numbers of objects stabilizes. Here we chose an optimum value of $\beta = 45$. Considering other frames from different shots might be having objects of size greater than 45 and less than 50.

Tolerance meter preset α	Size of the Noise object.	Total number of objects
6	----	>50
7	8580	26
8	6270	10
9	4855	10
10	3940	12
11	3258	12
12	2721	13
13	2326	11
14	2105	9
15	1825	9
16	1608	7
17	1413	7
18	1238	8
19	1125	7
20	943	5

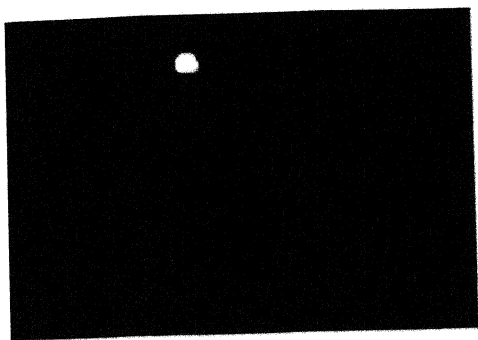
Table 5.1 The results with various value of α

Tolerance meter preset β	Size of the noise object	Total number of objects
20	1767	11
25	1767	11
30	1795	10
35	1825	9
40	1825	9
45	1825	9
50	1825	9

Table 5.2 The results with various values of β

After setting these two parameters, for the image 136x200 "tennis.002.tif" the identified objects are shown in the next page.

The Fourth Object



The Fifth Object

The Sixth Object



The Seventh Object



The Eighth Object

Fig 5.2 Identified objects.

Step 2

Set the value of γ , δ and μ for any two consecutive frames for a shot. Now to set the values of γ and δ we have chosen the first two frames from the first shot.

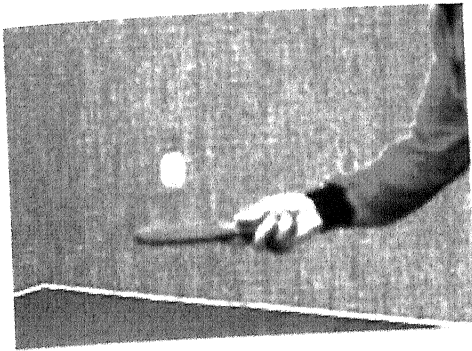
With the previously set α and β the objects are identified for two frames shown in the next page. Both the frames are having eight objects including the noise object. Now as discussed in Section 3, the distance matrices between these two frames are given in table 5.3 and 5.4. For the intensity distance we set the value of $\gamma=20$ and for the size distance we set the value of $\delta=100$. This value depends on the variation of objects in size from frame to frame. This much of relaxation is given considering the frames of other shots.

Objects	First	Second	Third	Fourth	Fifth	Sixth	Seventh
First	1	43	73	142	78	132	112
Second	42	0	65	144	111	96	106
Third	67	62	66	179	132	125	46
Fourth	157	159	194	15	79	225	235
Fifth	33	28	46	161	98	105	78
Sixth	131	96	59	213	200	1	112
Seventh	111	106	54	223	176	113	1

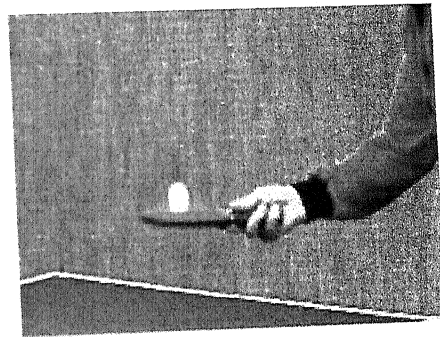
Table 5.3 The distance matrix considering intensity.

Objects	First	Second	Third	Fourth	Fifth	Sixth	Seventh
First	27	17751	20149	20145	20057	20018	18088
Second	17789	11	2387	2383	2295	2256	326
Third	19997	2219	179	175	87	48	1882
Fourth	20210	2432	34	38	126	165	2095
Fifth	20199	2421	23	27	115	154	2084
Sixth	20045	2267	131	127	39	0	1930
Seventh	18147	369	2029	2025	1937	1898	32

Table 5.4 The distance matrix considering size.



First Frame



Second Frame

Fig5.3 Two consecutive frames used for setting the value of γ and δ

Objects of First Frame	Objects of Second Frame
First Object	First Object
Second Object	Second Object
Fourth Object	Fourth Object
Sixth Object	Sixth Object
Seventh Object	Seventh Object

Table 5.5 The established correspondence

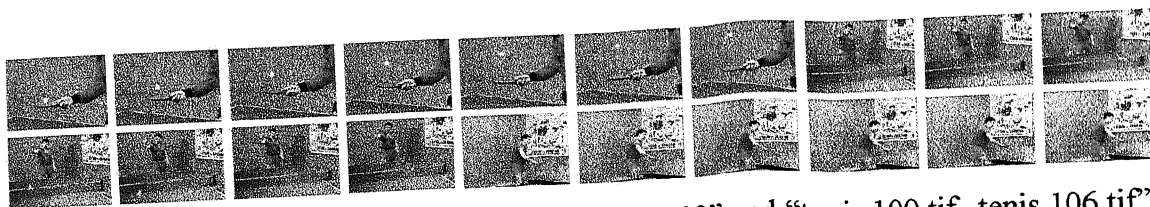
μ = The tolerance meter preset for the tolerance meter provided to take a decision.

To set the value of μ , we first calculated the extent of image match in terms of number of pixels. Here it is the total size of the matched objects. And it is 25165. Total image size is $136 \times 200 = 27200$. So the percentage of unmatched part is 7.5 %. Here we set the value of $\mu = 15\%$. So if 85 % of the image is matched then there is no shot change.

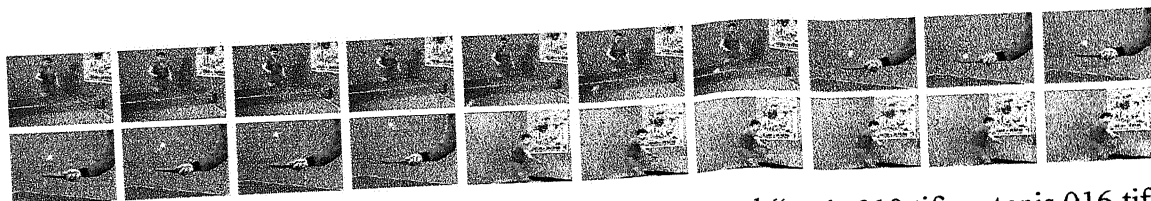
The value of γ and δ may need to modify later considering the other frames taken from different shot.

To make a decision as to whether a major object appeared or not, first we checked whether there had been a shot change. If both the frames are from same shot, then we calculate the total size of the major objects (objects with size larger than a particular value say 1000.) in both the frames. Next we calculate the total size of the major *matched* objects. If the total major object size is much greater than the total *matched* major objects size of the previous frame, then there is reason to suspect that a major object disappeared: if the same happens for the subsequent frame (in relation to the current one) then a major object has appeared.

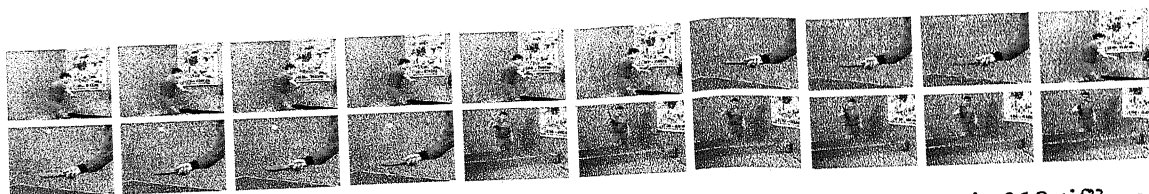
We have used the following test sequences:



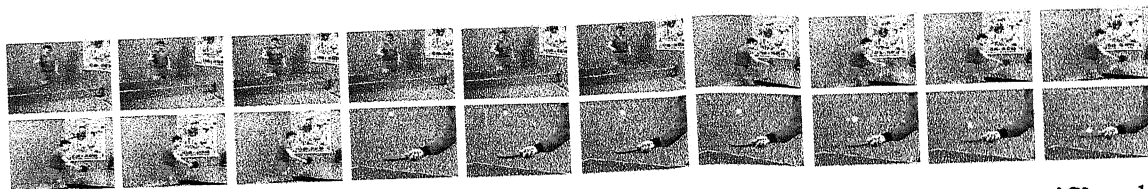
Sequence 1: (20 frames) "tenis.026.tif"---"tenis.032.tif" and "tenis.100.tif--tenis.106.tif" and "tens221.tif--tens226.tif".



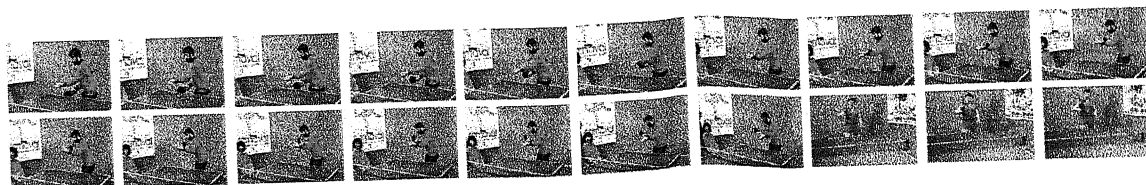
Sequence 2: (20 frames) "tenis.114.tif ---tenis.120.tif" and "tenis.010.tif --- tenis.016.tif" and "tens226.tif--tens231.tif".



Sequence 3: (20 frames) "tens231.tif--tens236.tif" and "tenis.016.tif--tenis.018.tif" and "tens237.tif" and "tenis.019.tif--tenis.022.tif" and "tenis.106.tif---tenis.111.tif".



Sequence 4: (20 frames) "tenis.110.tif---tenis.115.tif" and "tens237.tif--tens243.tif" and "tenis.020.tif--tenis.026.tif"



Sequence 5: (20 frames) "ten44.tif---ten60.tif" and "tenis.110.tif--tenis.112.tif"

The results of the algorithm using the aforementioned sequences are shown in tabular form in the subsequent pages. The values of tolerance meters preset are

$$\gamma = 20, \delta = 300, \text{ and } \varphi = 30.$$

	Frames	Shot status	Appearance of objects
1	"tenis.026.tif"	-----	-----
2	"tenis.027.tif"	Same Shot	Same number of major objects
3	"tenis.028.tif"	Same Shot	Same number of major objects
4	"tenis.029.tif"	Same Shot	Same number of major objects
5	"tenis.030.tif"	Same Shot	Same number of major objects
6	"tenis.031.tif"	Same Shot	Same number of major objects
7	"tenis.032.tif"	Same Shot	Same number of major objects
8	"tenis.100.tif"	Shot Changed	-----
9	"tenis.101.tif"	Same Shot	Same number of major objects
10	"tenis.102.tif"	Same Shot	Same number of major objects
11	"tenis.103.tif"	Same Shot	Same number of major objects
12	"tenis.104.tif"	Same Shot	Same number of major objects
13	"tenis.105.tif"	Same Shot	Same number of major objects
14	"tenis.106.tif"	Same Shot	Same number of major objects
15	"tens221.tif"	Shot Changed	-----
16	"tens222.tif"	Same Shot	Same number of major objects
17	"tens223.tif"	Same Shot	Same number of major objects
18	"tens224.tif"	Same Shot	Same number of major objects
19	"tens225.tif"	Same Shot	Same number of major objects
20	"tens226.tif"	Same Shot	Same number of major objects

Table 5.6. Change detection of Sequence 1.

	Frames	Shot status	Appearance of objects
1	"tenis.114.tif"	-----	-----
2	"tenis.115.tif"	Same Shot	Same number of major objects
3	"tenis.116.tif"	Same Shot	Same number of major objects
4	"tenis.117.tif"	Same Shot	Same number of major objects
5	"tenis.118.tif"	Same Shot	Same number of major objects
6	"tenis.119.tif"	Same Shot	Same number of major objects
7	"tenis.120.tif"	Same Shot	Same number of major objects
8	"tenis.010.tif"	Shot Changed	-----
9	"tenis.011.tif"	Same Shot	Same number of major objects
10	"tenis.012.tif"	Same Shot	Same number of major objects
11	"tenis.013.tif"	Same Shot	Same number of major objects
12	"tenis.014.tif"	Same Shot	Same number of major objects
13	"tenis.015.tif"	Same Shot	Same number of major objects
14	"tenis.016.tif"	Same Shot	Same number of major objects
15	"tens226.tif"	Shot Changed	-----
16	"tens227.tif"	Same Shot	Same number of major objects
17	"tens228.tif"	Same Shot	Same number of major objects
18	"tens229.tif"	Same Shot	Same number of major objects
19	"tens230.tif"	Same Shot	Same number of major objects
20	"tens231.tif"	Same Shot	Same number of major objects

Table 5.7. Change detection of Sequence 2.

	Frames	Shot status	Appearance of objects
1	"tens.231.tif"	-----	-----
2	"tens.232.tif"	Same Shot	Same number of major objects
3	"tens.233.tif"	Same Shot	Same number of major objects
4	"tens.234.tif"	Same Shot	Same number of major objects
5	"tens.235.tif"	Same Shot	Same number of major objects
6	"tens.236.tif"	Same Shot	Same number of major objects
7	"tenis.016.tif"	Shot changed	-----
8	"tenis.017.tif"	Same Shot	Same number of major objects
9	"tenis.018.tif"	Same Shot	Same number of major objects
10	"tens.237.tif"	Shot changed	-----
11	"tenis.019.tif"	Shot Changed	-----
12	"tenis.020.tif"	Same Shot	Same number of major objects
13	"tenis.021.tif"	Same Shot	Same number of major objects
14	"tenis.022.tif"	Same Shot	Same number of major objects
15	"tenis.106.tif"	Shot Changed	-----
16	"tenis.107.tif"	Same Shot	Same number of major objects
17	"tenis.108.tif"	Same Shot	Same number of major objects
18	"tenis.109.tif"	Same Shot	Same number of major objects
19	"tenis.110.tif"	Same Shot	Same number of major objects
20	"tenis.111.tif"	Same Shot	Same number of major objects

Table 5.8. Change detection of Sequence 3.

	Frames	Shot status	Appearance of objects
1	"tenis.110.tif"	-----	-----
2	"tenis.111.tif"	Same Shot	Same number of major objects
3	"tenis.112.tif"	Same Shot	Same number of major objects
4	"tenis.113.tif"	Same Shot	Same number of major objects
5	"tenis.114.tif"	Same Shot	Same number of major objects
6	"tenis.115.tif"	Same Shot	Same number of major objects
7	"tens237.tif"	Shot Changed	-----
8	"tens238.tif"	Same Shot	Same number of major objects
9	"tens239.tif"	Same Shot	Same number of major objects
10	"tens240.tif"	Same Shot	Same number of major objects
11	"tens241.tif"	Same Shot	Same number of major objects
12	"tens242.tif"	Same Shot	Same number of major objects
13	"tens243.tif"	Same Shot	Same number of major objects
14	"tenis.020.tif"	Shot Changed	-----
15	"tenis.021.tif"	Same Shot	Same number of major objects
16	"tenis.022.tif"	Same Shot	Same number of major objects
17	"tenis.023.tif"	Same Shot	Same number of major objects
18	"tenis.024.tif"	Same Shot	Same number of major objects
19	"tenis.025.tif"	Same Shot	Same number of major objects
20	"tenis.026.tif"	Same Shot	Same number of major objects

Table 5.9. Change detection of Sequence 4.

	Frames	Shot status	Appearance of objects
1	"ten44.tif"	-----	-----
2	"ten45.tif"	Same Shot	Same number of major objects
3	"ten46.tif"	Same Shot	Same number of major objects
4	"ten47.tif"	Same Shot	Same number of major objects
5	"ten48.tif"	Same Shot	Same number of major objects
6	"ten49.tif"	Same Shot	Same number of major objects
7	"ten50.tif"	Same Shot	Same number of major objects
8	"ten51.tif"	Same Shot	Same number of major objects
9	"ten52.tif"	Same Shot	Same number of major objects
10	"ten53.tif"	Same Shot	Same number of major objects
11	"ten54.tif"	Same Shot	Same number of major objects
12	"ten55.tif"	Same Shot	New major object appeared
13	"ten56.tif"	Same Shot	Same number of major objects
14	"ten57.tif"	Same Shot	Same number of major objects
15	"ten58.tif"	Same Shot	Same number of major objects
16	"ten59.tif"	Same Shot	Same number of major objects
17	"ten60.tif"	Same Shot	Same number of major objects
18	"tenis.110.tif"	Shot Changed	-----
19	"tenis.111.tif"	Same Shot	Same number of major objects
20	"tenis.112.tif"	Same Shot	Same number of major objects

Table 5.10. Change detection of Sequence 5.

For sequence 5 any object of size greater than 450 is considered as a major object.

Section 6

Conclusion

In this thesis report we discussed the application of tolerance relations for various image operations such as representation, image and video segmentation and texture analysis. In the representation problem, it was found that the representation was not unique, unless a sufficiently high order tolerance meter is applied.

The image and video segmentation problems were handled much more satisfactorily. The texture analysis carried out using tolerance devices also yielded results. However, the choice of the parameters requires manual intervention.

While there are many shortcomings yet to be overcome, we can claim to have made a beginning at understanding some aspects of the use of tolerance spaces for signal processing operations. Essentially, it may be said that the notion of a tolerance provides novel ways to come to decisions on many questions based on information provided by features.

The work we have done can be extended further in the analysis of textural objects. Change detection and change blindness problems of other kinds than those we have attempted here would generally find our approach useful. There are also some other possibilities.

1. Tolerance relation can be used to remove speckles from an image.
2. For indexing purposes, where classification requires, tolerance relation can be used as a classifier. Wherever there is a problem of classification arises tolerance relation can be used there as a classifier.

References

- [1] Ju. A. Schreider, "Equality, Resemblance, and order.", Mir Publishers, 1975.
- [2] A. K. Jain, "Fundamentals of Digital Image processing", Prentice Hall, 1988.
- [3] R.M. Haralick, K. Shanmugam, and I. Dinstein, "Textural Features for Image Classification." *IEEE Trans. System Man Cybern*, vol. smc-3, no. 6, pp. 610-621, November 1973.

Errata

Page No.	Line No.	
1. 7	2	“ It is” → “Its”.
2. 9	10	“Tolerances” → “Tolerance”.
3. 75	10	“Same no. of object” → “New object appeared”.
4. 75	11	“Same no. of object” → “New object appeared”.
5. 75	12	“New object appeared”→ “Same no. of objects”.
6. 75	17	“Same no. of object” → “Object disappeared”.
7. 75	19	“Same no. of object” → “Object disappeared”.
8. 76	19	“arises” will be removed.

A

139574

